Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes

Treball de final de carrera

Disseny i control d'un robot equilibrista sobre dues rodes

Autor: Santi Galindo Nadal
Director: Ferran Perdrix Sapiña

21 de desembre de 2012

Índex

Llista de Figures

Llista de Taules

Agraïments

En aquest punt, m'agradaria donar les gràcies a totes les persones que han contribuït directa o indirectament en aquest treball.

En primer lloc agrair al Txema, qui em va introduir al mon de la robòtica i qui ha estat en tot moment al meu costat, ensenyant-me i guiant-me en aquest camí tant complex que jo desconeixia, i que gràcies al seu temps i paciència ara conec una mica més.

Al Ferran, que com a director de projecte va confiar en mi per tal de dur a terme aquest treball. També l'hi he d'agrair el seu suport i per ajudar a posar en ordre les idees, estructurar aquest treball, i també pels seus valuosos comentaris, revisions i suport al llarg de tot aquest temps.

A tots aquells amics i companys que m'han fet costat al llarg de tots aquests anys i que m'han fet veure que hi ha un altre món a part de la universitat.

I sobretot als meus pares, la meva germana, la Txaro i el Jordi, per donar-me suport incondicional i continu, escoltar-me i animar-me sempre. Amb ells he compartit inquietuds, dubtes, nervis i també bons moments que ho han fet tot sempre més fàcil. Ells han fet aquest projecte possible.

A tots aquells que d'una manera o altra m'han ajudat, encara que el seu nom no figuri de forma explícita en aquestes línies; sense ells tampoc hauria estat possible.

Gràcies a tots.

Capítol 1

Introducció

Des de l'inici dels temps, l'home ha desitjat crear vida artificial. Ha volgut donar vida a éssers artificials que els ajudin, éssers que realitzin feines repetitives, feines pesades o difícils de realitzar per l'home. Alguns autors, com J.J.C. Smart (filòsof australià i professor) i Jasia Reichardt (crítica d'art i editora), consideren què el primer autòmat en la història va ser Adan creat per Déu. D'acord amb això, Adan i Eva són els primers autòmats intel·ligents creats, i Déu va ser qui els va programar i els va donar les primeres instruccions que havien de seguir. Dins de la mitologia grega trobem alguns relats sobre la creació de vida artificial, per exemple, Prometéus va crear el primer home i la primera dóna amb fang i els va animar amb el foc del cel. D'aquesta manera ens adonem que la humanitat té l'obsessió de crear vida artificial des de el principi dels temps. Molts han estat els intents per aconseguir-ho.

Però en els últims anys, la tecnologia cada vegada està més present en les nostres vides quotidianes. Això ha provocat un augment de l'intérès per automatitzar i facilitar, no només ens el sector empresarial, sinó també en el sector dels serveis. Amb totés aquestes necessitats, la robòtica ha crescut exponencialment. Tenim robots que busquen vida intel·ligent en altres planetes (1.0.1), robots amb utilitats militars (1.0.2), o fins i tot, aspiradores domèstiques autònomes (1.0.3).

Els robots es divideixen en estàtics i dinàmics. Un dels robots dinàmics més famosos són els anomenats pènduls invertits de dos rodes. La millor manera d'entendre el funcionament del pèndul invertit és imaginar-nos que és un joc. La majoria de persones han jugat a aguantar amb la palma de la mà una escombra invertida. La idea del joc és no deixar caure l'escombra, per tant, hem d'estar pendents per actuar ràpidament davant qualsevol inclinació d'aquesta. En el moment que observem que cau cap una banda o l'altra, el jugador mou ràpidament la seva mà intentant estabilitzar l'escombra esperant que aquesta es mantingui el més quieta possible en posició vertical invertida. Aquest joc es coneix en la teoria del control modern com un pèndul invertit. El pèndul invertit no es més que una barra (l'escombra en el nostre joc), la qual s'ha de mantenir en posició vertical, sent el seu punt de suport l'extrem inferior. On l'estabilitat es donada gràcies a una acció de control, en el cas del joc: el moviment de la mà.

Encara que el joc de l'escombra sembla fàcil, com a teoria de control no ho és. Tot sistema de control, sigui quina sigui la seva aplicació està dividit en tres passos: mesurar, controlar i actuar. En el cas de l'escombra, la persona mesura quan observa amb els seus ulls que l'escombra tira cap una banda. Per tant, pren una decisió de control, la qual indica a la mà que s'ha de traslladar cap a una banda perquè l'escombra no caigui a terra. Aquestes tres accions, són utilitzades a diari per l'ésser humà en totés les activitats de la vida.

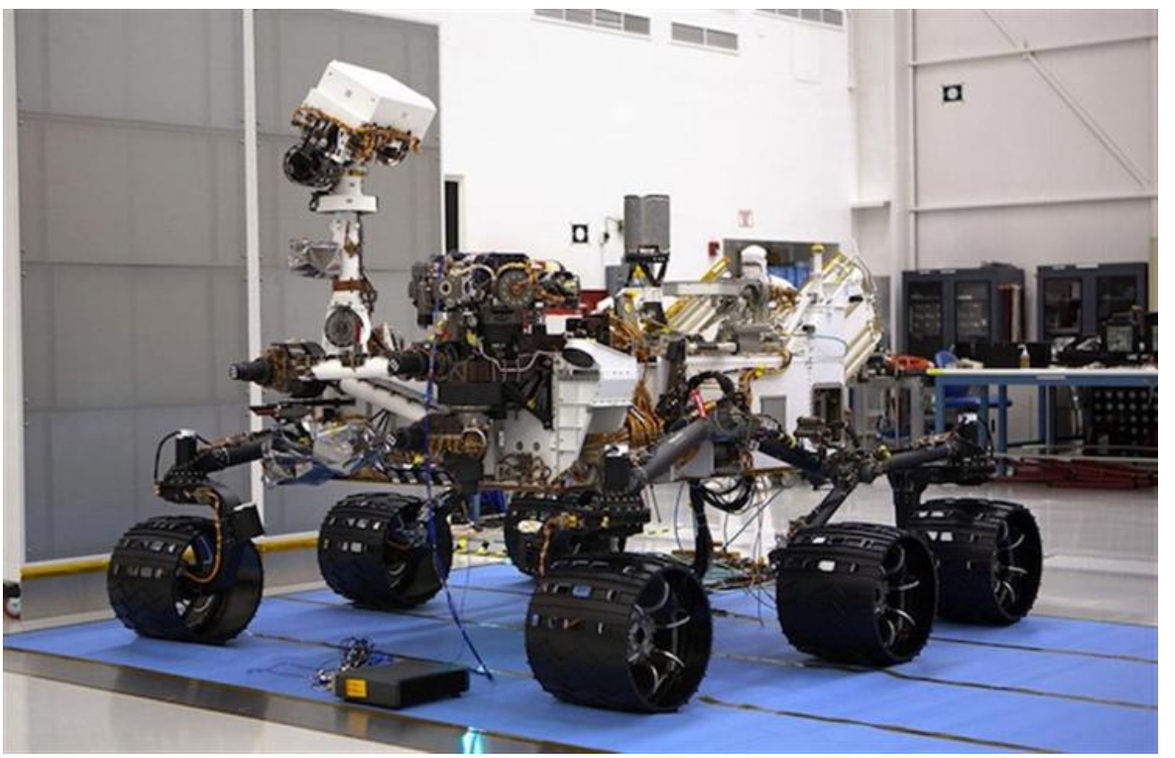


Figura 1.0.1: Curiosity

Figura 1.0.2: Bigdog

Figura 1.0.3: Roomba

Capítol 2

Motivació

La robòtica és una nova tecnologia que va sorgir sobre el 1960. En pocs anys s'han produït grans avanços i el més important és que la robòtica ha provocat un gran interès, fet que fa que estigui en continua evolució.

Durant els últims anys, trobem que la tecnologia està més present en la vida de les persones. Això ha provocat que la robòtica tingui un paper molt important a nivell professional i també a nivell particular.

La robòtica és una de les parts més importants i complicades de la enginyeria en la que diferents disciplines aporten el seu gra d'arena per poder realitzar un projecte molt ambiciós.

Una de les principals diferències que hem de destacar en els robots, mecànicament parlant, és el seu tipus d'estabilitat. Podem trobar robots d'estabilitat estàtica i d'estabilitat dinàmica. Diem que un robot és d'estabilitat estàtica quan el seu funcionament no afecta al seu centre de gravetat. Aquest és el cas dels robots que van amb 3 o 4 rodes.

En el meu projecte he apostat per l'estudi i realització d'un robot equilibrista d'estabilitat dinàmica.

Els robots d'aquesta classe són més complicats des del punt de vista del disseny i del control. Però tenen algunes avantatges més destacades com la capacitat d'evitar obstacles, la capacitat de moure's per terrenys més inestables, etc .

Una altra de les característiques d'aquests robots és que són més humanoides, ja que la tendència en el món de la robòtica és que tinguin cada vegada més similitud a nosaltres i puguin realitzar feines més humanes.

En el meu projecte he elegit construir un robot equilibrista i donar-li una utilitat. El fet que estigui construït sobre dos rodes fa que es comporti com un pèndul invertit. Un dels robots més famosos que es comporta com un pèndul invertit es el Segway (imatge 2.0.1) , en el qual m'he inspirat.

Per saber més, mirar [PAL08, SIL07].

Figura 2.0.1: Segway i2

Capítol 3

Objectius

1. L'objectiu d'aquest PFC és construir un robot per empreses capaç de transportar diferents tipus de documents, com el correu en una empresa, la carta de menús en un restaurant o la beguda en un bar. El robot es guiarà a través d'uns sensors que llegiran unes línies al terra. També es podrà controlar via Bluetooth a través de la seva aplicació Java.

2. Es buscarà construir un robot de manera modular, es a dir, que ens permeti la substitució, eliminació i incorporació de nous mòduls sense tenir que modificar la resta de mòduls que componen el robot.

3. El principal objectiu del projecte és aconseguir un sistema de control que permeti que el robot estigui en equilibri sobre dues rodes, fent que no es desplaci. Per poder-me centrar en què només realitzés la seva feina, he dissenyat tot el robot: des de la seva PCB fins a l'aparença física del robot.

4. És voldran assolir tres fases, la primera serà el disseny i construcció de la PCB i de la part física. La segona fase serà la programació del microcontrolador. És buscarà poder fer que es mantingui en equilibri i realitzi les altres tasques secundàries.

5. I la tercera i última fase, serà la programació d'una aplicació Java per poder comunicar-me amb el robot, on podrem llegir les dades per veure el seu estat i enviar ordres de moviment i ordres d'ajust.

6. Finalment, com a objectius addicionals, s'han integrat 3 funcions més al robot. Primerament, és voldrà integrar un control remot per Buetooth a partir de l'aplicació mencionada al paràgraf anterior. També un sistema anticolisions, utilitzant un sensor d'ultrasons. I per acabar, com que la utilitat del robot serà portar documents/cartes dins d'una empresa, s'adaptarà un sistema IR per saber si el robot està transportant alguna cosa.

7. L'últim objectiu d'aquest treball és documentar la feina realitzada per tal que en quedi constància i faciliti futures modificacions.

Capítol 4

Estructura de la memòria

En aquest document escriurem les diferents fases per poder construir un robot equilibrista i li donarem un ús.

Primerament descrivim què és un robot equilibrista i com funciona, també veurem quins usos pot arribar a tenir i quines funcionalitats tenen el robots equilibristes que es troben en el mercat avui en dia.

En aquest punt, vist les funcionalitats que trobem en els robots ja en el mercat, dissenyem i construim el nostre robot, seguint unes pautes establertes perquè pugui realitzar les funcions descrites en els objectius.

Finalment, comencem a donar vida al robot, programant els diferents sensors perquè pugui arribar a desenvolupar les funcionalitats per exercir com a robot equilibrista i com a robot transportador. També expliquem les eines utilitzades per poder programar-lo. Així com el protocol de comunicació per poder comunicar-nos amb ell.

Un cop tinguem el robot amb tots els seus sentits operatius, tanquem aquest apartat amb una utilitat, en el cas del nostre projecte s'utilitza llenguatge Java. Aquesta utilitzat ens permet entendre aquest protocol de comunicació i ens permet tenir una telemetria per poder ajustar i enviar dades addicionals al robot.

En els últims apartats del projecte trobem els resultats obtinguts, així com un manual d'usuari, seguit de les conclusions, les futures extensions i un annex amb els datasheets dels sensors utilitzats.

Capítol 5

Estat de l'art

Actualment existeixen varis projectes importants que estan centrats en aquest tipus de solució. Si busquem en el mercat un robot equilibrista que utilitza la tècnica del pèndul invertit el primer que trobem es el famós Segway.

Dean Kamen va meravellar el món el desembre de 2001 amb el seu Segway Personal Transport. Segway va ser el primer transport humà auto-equilibrat. Els sensors, microcontrolador i motors situats a la base del Segway fan que aquest es mantingui en horitzontal tot el temps. Hi ha dos models bàsics de Segway, el i2 (imatge 2.0.1) i el x2 (imatge 5.0.1). El i2 es pot adaptar a qualsevol tipus de terreny i manté l'usuari sempre vertical, mentre que el x2 és un model més pesat destinat al ús per off-road.

El Segway i2 té un pes de 47kg, mentre que el Segway x2 té un pes de 54,4 kg. Els dos porten una bateria de ió-liti Saphion, en el model i2 aquestes bateries permeten recórrer 39 kms amb una sola carga, en canvi en el model x2 li permeten recórrer 20 kms, ja que està preparat per altres tipus de térreny que fa que la bateria s'acabi més ràpidament.

Tècnicament, el Segway compta amb cinc sensors giroscopis, tot i que només utilitza tres, els altres sensors són redundants i incrementen la seguretat i fiabilitat.

El cervell i la força són responsabilitat de dos grups redundants de components compostos per microprocessadors, un control elèctric, bateries i motors. En el cas que falli un dels molts components, s'activarà l'altre component que funciona com a mirall, el qual assumeix instantàniament el control de les seves funcions, així l'usuari no està mai en perill.

La creació de Dean Kamen només ha sigut la punta del iceberg, molts han sigut els que han agafat la idea de Dean i han adaptat les seves idees en el seu Segway. Veurem un parell d'exemples: Steadycam en un Segway i Genny.

La empresa Handsfree-transporter es dedica a construir y comercialitzar vehicles Segway "mans lliures", pensats principalment per ser utilitzat juntament amb una steadycam (imatge 5.0.2). El fet de poder anar de forma estable i a una velocitat considerable per llocs d'impossible accés per una furgoneta o fins i tot una moto, per fer tràvelings de seguiment, dóna un potencial increïble. Aquests aparell ha estat utilitzat en pel·lícules de Martin Scorsese, i també l'hem pogut veure en les últimes olimpíades.

Com a característica destacada podem dir que el pes és de 50kg, un pes més que bo, considerant que una moto o una furgoneta pesen bastant més.

Figura 5.0.1: Segway x2

Figura 5.0.2: Steadycam

Figura 5.0.3: Geanny 2.0

Un altre projecte és Genny (imatge 5.0.3). Una creació de l'italià Paolo Badano. Fa uns anys, el curiós mitjà de transport elèctric que es deia Segway l'hi va cridar l'atenció, tenia la funcionalitat i la capacitat d'estabilització que la convertien en una màquina màgica. Però per Paolo Badano tenia un problema, i era que per utilitzar-lo era necessari posar-se de peu en posició perfectament vertical, ja que Paolo Badano va tenir un accident que el va obligar a aprendre a conviure en una cadira de rodes. Però ell va pensar que si Segway era capaç de mantenir en equilibri de manera segura una persona amb el centre de gravetat molt allunyat del terra també podia fer el mateix amb una persona assentada. Així doncs, amb unes modificacions del Segway i molt treball va néixer Genny 2.0.

Com podem veure Segway ha estat la inspiració de moltes persones. Un altre exemple és el realitzat per un company i amic de Fraga, que va decidir fer un Segway de baix cost (imatge 5.0.4). També existeixen variacions, com es el cas del projecte Rezero (imatge 5.0.5)(per saber més, mirar [FON09]), creat per tretze enginyers d'universitats diferents de Suïssa. El seu robot utilitza 3 rodes mini direccionals per actuar sobre una esfera. Un sistema que ofereix una gran estabilitat i control del robot. Algunes de les funcions que té el robot són el control d'estabilitat i trajectòria, sistema de seguiment làser, orbitació al voltant d'un objecte i control remot del robot. A més s'ha inclòs una etapa molt important de l'electrònica de potència per poder alimentar motors de gran par que fa que el robot sigui molt robust davant variacions de pes i de terreny.

També existeixen diferents tipus de giroscopis. Els més utilitzats eren els mecànics, però degut al volum i el pes d'aquests van començar a tornar-se incompatibles per moltes aplicacions, així doncs van sortir els electrònics. De giroscopis electrònics n'hi ha de varis tipus depenent del seu efecte. Trobem així tipus com el piezoelèctrics, que funcionen a partir d'un cristall que oscil·la i aquest varia la seva resistència elèctrica, els capacitatius, que fan els càlculs a partir de la diferència de capacitat a partir de la velocitat de rotació que porta el giroscopi, o els MEMS, que veurem més endavant ja que és el tipus que utilitzarem en el projecte.

En el cas de l'acceleròmetre també existeixen diferents tipus. Com els mecànics, amb el mateix problema que els giroscopis mecànics, els capacitatius, que modifiquen la posició relativa de les plaques d'un micro-condensador quan està sotmès a una acceleració, els piezoelèctrics, que es basen en l'efecte piezoelèctric i els MEMS, utilitzat també en el projecte.

Trobem també diferents tipus de mòduls Buetooth, aquests estan dividits en 3 classes segons el seu radi de cobertura. Així doncs podem trobar al mercat:

- Classe 1: Pot transmetre a distàncies de fins a 100m

Figura 5.0.4: Segway baix cost

- Classe 2: Pot transmetre a distàncies de fins a 50m
- Classe 3: Pot transmetre a distàncies de fins a 10m

De tipus de microcontroladors tenim varis models en el mercat, però per aquest tipus d'aplicació dos són els més importants. Trobem els PIC i els Atmel. Els dos estan molt extensos en el món de la robòtica i realment no hi ha un clar dominant, simplement la raó per decidir-se entre un o l'altre és l'experiència o la curiositat per saber el funcionament dels dos.

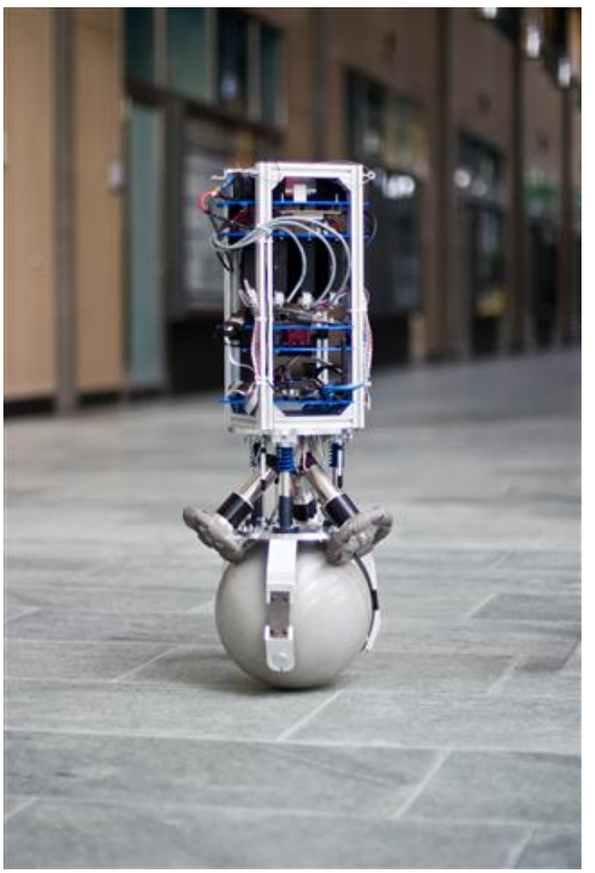


Figura 5.0.5: Rezero

Capítol 6

Tecnologies Utilitzades

6.1. Programes

6.1.1. Programmer's Notepad (Win AVR)

WinAVR és una aplicació que ens dóna la possibilitat de desenvolupar firmwares per microcontroladors AVR, aquesta aplicació es basa en el conegut compilador GCC que és de codi lliure i obert. A més a més, tenim molta informació que està disponible a la WEB. A l'instal·lar aquest aplicació, s'instal·len també les llibreries per poder treballar en microcontroladors AVR, una versió del compilador GCC, i algunes altres aplicacions.

Després d'instal·lar les llibreries amb el seu compilador, ja no ens queda cap més remei que escriure algunes línies i fer-ho córrer.

En la figura 6.1.1 veiem un exemple de la interfície.

6.1.2. Flip 3.4.5 (FLexible In-systém Programmer)

Flip és un bootloader per a Atmel. Amb el programa Flip podem programar un AVR que tingui carregat prèviament un ISP via hardware. En el meu cas ja tenia aquest ISP prèviament programat, el que el permet carregar qualsevol firmware sense necessitat de tenir un programador extern per Atmel.

Això només és possible pels Atmels amb USB, en el meu cas, el Atmega 32u4 està preparat per això.

Amb aquest programa també podem veure informació extra del microcontrolador, com l'espai utilitzat, la memòria lliure, o els registres utilitzats. Fins i tot eliminar qualsevol firmware carregat prèviament en el Atmel.

En la figura 6.1.2 veiem un exemple de la interfície.

6.1.3. Advanced Serial Port Terminal

Advanced Serial Port Términal és una aplicació que proporciona una intérface de comunicació simple per connectar-se a qualsevol dispositiu de port sèrie, com podria ser un mòdem, fax... o en el nostre cas un dispositiu Buetooth preparat per funcionar amb aquest protocol.

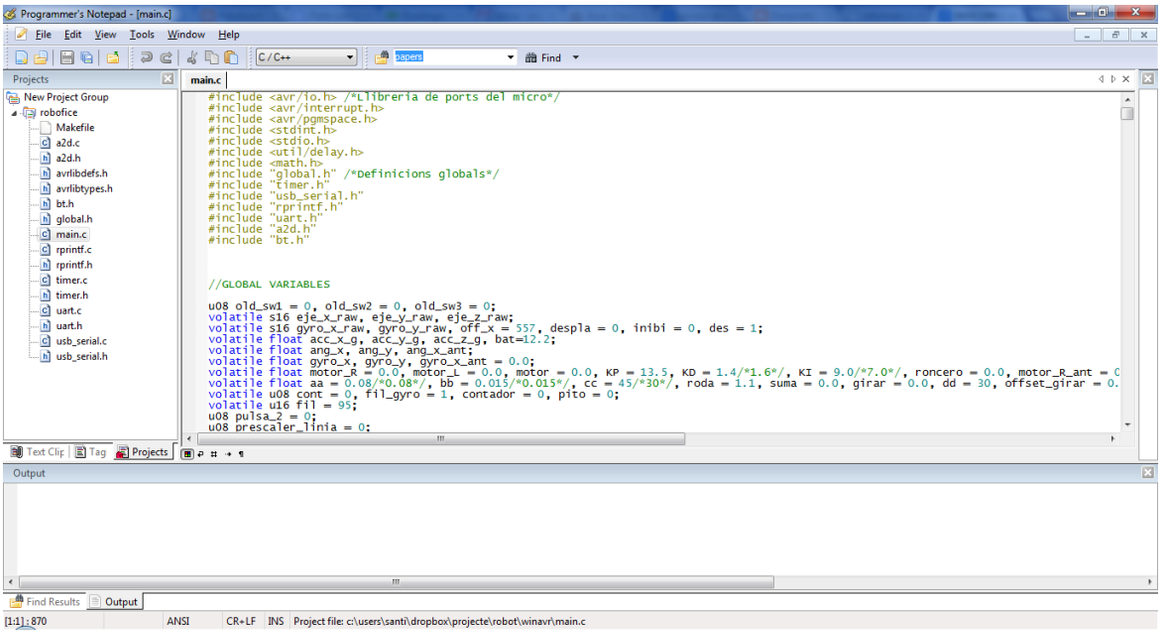


Figura 6.1.1: Exemple interfície Programmer's Notepad (Win AVR)

Al ésser una solució basada en sessió, ens permet crear conjunts d'opcions i guardar-les per utilitzar en altres sessions. També ens permet guardar sessions, cosa que he utilitzat per comparar diferents resultats dels sensors en diferents configuracions.

Una de les opcions que més he utilitzat és la possibilitat de rebre i enviar dades en format hexadecimal, així com la possibilitat d'associar a les tecles diferents comandes ràpides, aquesta última em va ajudar molt a l'hora de fer les proves pel control remot del robot.

En la figura 6.1.3 veiem un exemple de la interfície.

6.1.4. Netbeans

Netbéans és un entorn de programació integrat lliure on existeixen un nombre important de mòduls per estendre'l. La plataforma NetBéans permet desenvolupar diferents aplicacions a partir de components de software, que són els mòduls. En el cas del meu projecte, he utilitzat netbeans 7.1.2 per fer una utilitat de comunicació i control des de el pc. Utilitzant la llibreria RXTXcomm.jar per poder llegir el protocol sèrie. Per fer l'apartat gràfic he utilitzat la biblioteca gràfica swing, ja que és la que està per defecte amb el compilador NetBéans.

En la figura 6.1.4 veiem una imatge del programa.

6.1.5. Google SketchUp

SketchUp és un programa de modelatge en 3D comercialitzat per Google i dissenyat per a arquitectura, enginyeria civil i enginyeria mecànica, així com animació i desenvolupaments de videojocs.

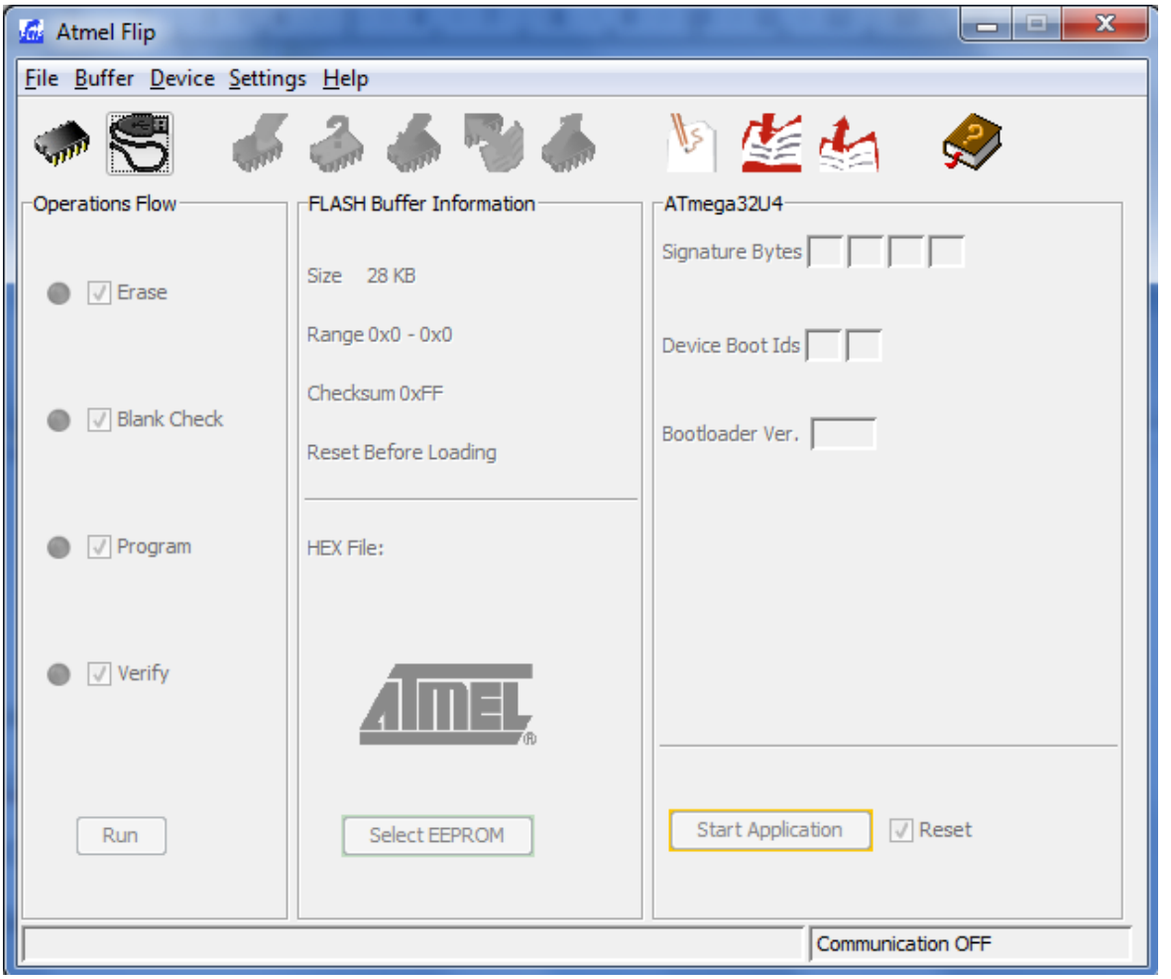


Figura 6.1.2: Exemple interfície Flip 3.4.5 (FLexible In-systém Programmer)

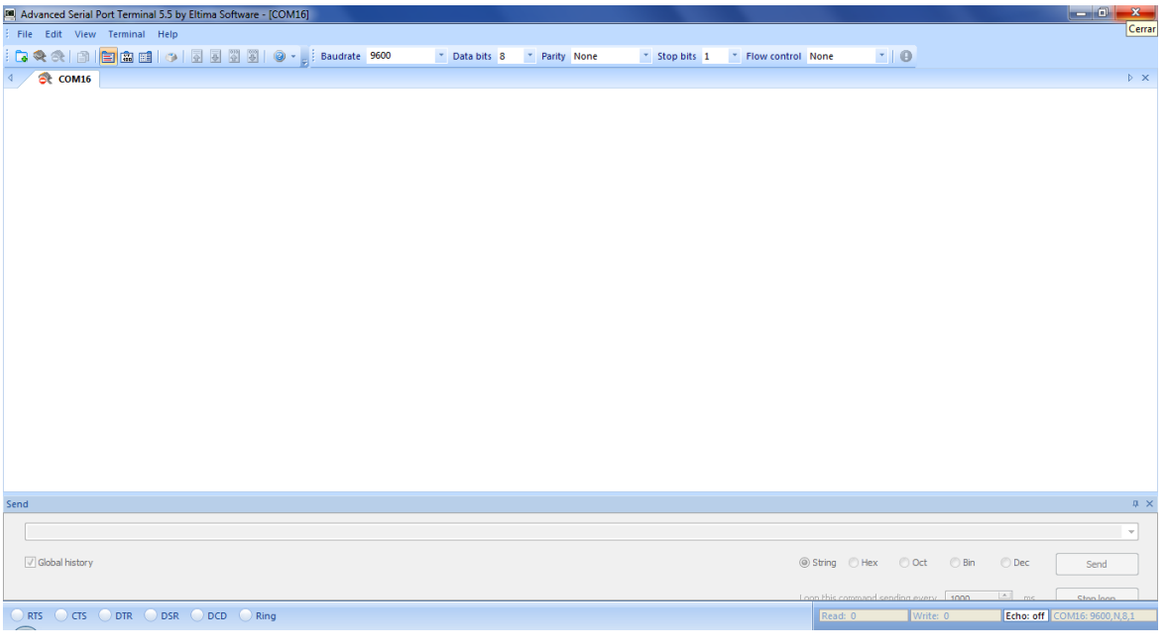


Figura 6.1.3: Exemple interfície Advanced Serial Port Terminal

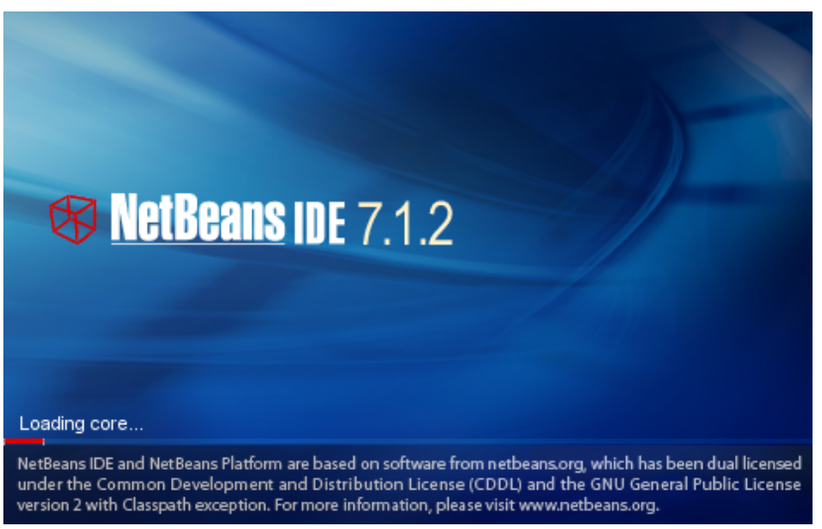


Figura 6.1.4: Exemple imatge Netbeans 7.1.2

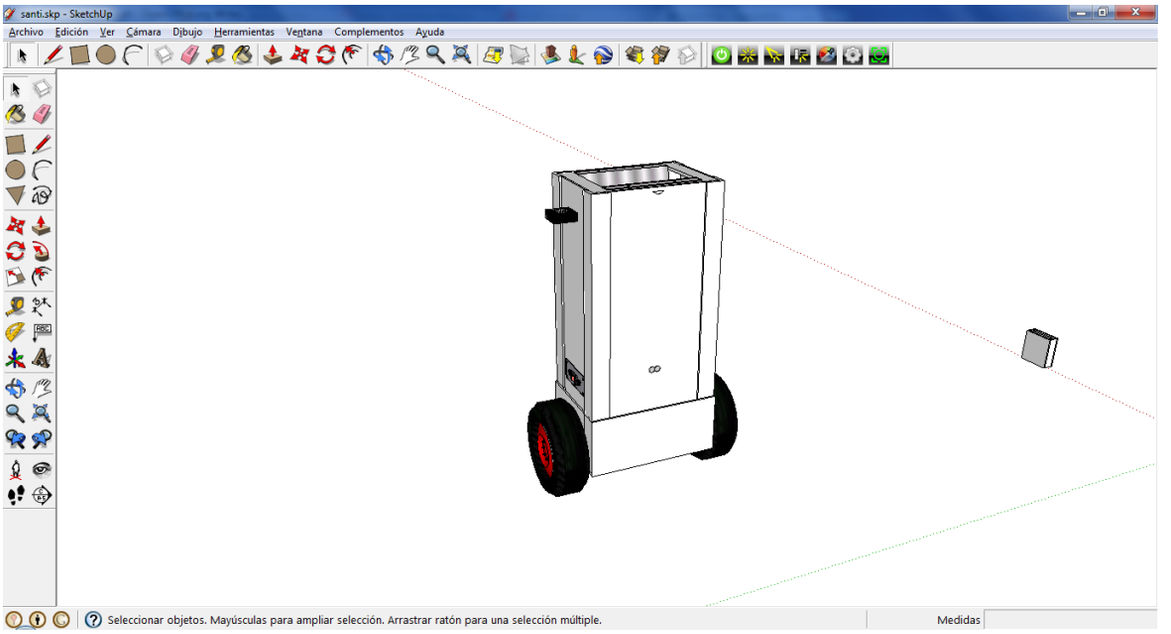


Figura 6.1.5: Exemple interfície Netbeans 7.1.2

En el cas del projecte, he utilitzat el Google SketchUp per dissenyar les peces que componen el robot i poder fer una simulació 3D del que seria el disseny final.

En la figura 6.1.5 veiem un exemple de la interfície.

6.1.6. Eagle 5.11.0

Eagle és una eina integral per desenvolupar circuits impresos, començant pels circuits elèctrics i plaques de circuits impresos. El programa implementa tres mòduls: Mòdul esquemàtic, editor de disseny i autorouter. A més a més, el programa té una gran biblioteca que conté una enorme varietat de components elèctrics bastant comuns i estàndards.

En aquest projecte he utilitzat Eagle per dissenyar la PCB a partir del mòdul esquemàtic i l'editor de disseny, preparant així el fotolit per poder fer la placa.

En la figura 6.1.6 veiem un exemple de la interfície.

Per saber més, mirar [EAG11].

6.2. Llenguatges emprats

6.2.1. Llenguatge C

El llenguatge C és un llenguatge de programació estructurat. El que significa que el codi o l'algoritme està ordenat o estructurat. Així doncs, és fàcilment diferenciable d'un llenguatge orientat a objectes.

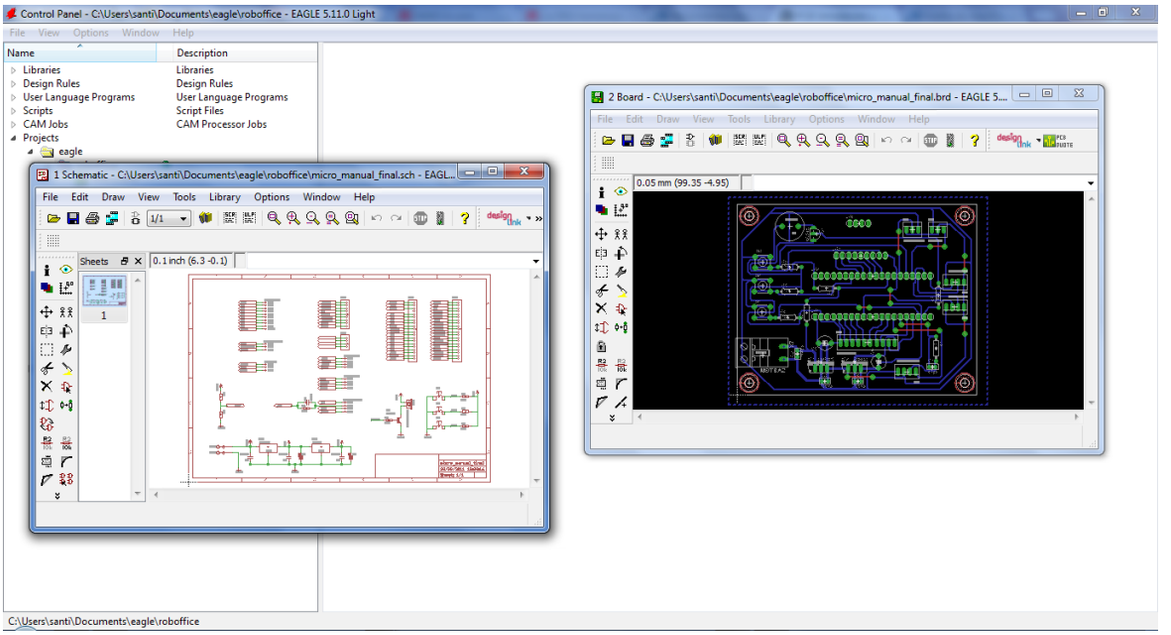


Figura 6.1.6: Exemple interfície Eagle 5.11.0

El llenguatge C és un dels llenguatges més ràpids i potents que existeixen avui en dia amb una sintaxis compacta i gran portabilitat.

En el projecte he utilitzat el llenguatge de programació C per escriure el firmware del microcontrolador Atmega32u4. El programa WinAVR ens permet utilitzar aquest llenguatge ja que utilitza el compilador GCC.

6.2.2. Llenguatge Java

Java és un llenguatge de programació orientat a objectes de propòsit general. Prové d'extensions que permeten del desenvolupament d'aplicacions gràfiques d'usuari (GUI), aplicacions multithreading, ús de multitud d'APIs estàndards, i és adequat per al desenvolupament d'aplicacions distribuïdes.

Java és un llenguatge de programació orientat a objectes desenvolupat per Sun Microsystéms a principis dels anys 90. El llenguatge en si mateix pren molta de la seva sintaxis de C i C++, però té un model d'objectes més simple i elimina eines de baix nivell, que solen induir a molts errors, com la manipulació directa de punters o memòria. Les aplicacions Java estan típicament compilades en un bytécode, encara que la compilació en codi màquina natiu també és possible. En el temps d'execució, el bytécode és normalment interpretat o compilat a codi natiu per a l'execució, encara que l'execució directa per maquinària del bytécode per un processador Java també és possible. La implementació original i de referència del compilador, la màquina virtual i les llibreries de classes de Java van ser desenvolupades per Sun Microsystéms en 1995. Des de llavors, Sun ha controlat les especificacions, el desenvolupament i l'evolució del llenguatge a través del Java Community Procéss, si bé uns altres han desenvolupat també implementacions alternatives d'aquestes tecnologies de Sun, algunes fins i tot sota llicències de programari lliure. Entre novembre de 2006 i maig de 2007, Sun Microsystéms va alliberar la major part de les seves tecnologies Java sota la llicència GNU, d'acord

amb les especificacions del Java Community Process, de tal forma que pràcticament tot el Java de Sun és ara programari lliure (encara que la biblioteca de classes de Sun que es requereix per a executar els programés Java encara no és programari lliure). Aquest és el llenguatge per al que m'he decidit per fer l'aplicació de comunicació i control des de el pc. Utilitzant la llibreria RXTXcomm.jar per poder llegir el protocol sèrie. Per fer l'apartat gràfic he utilitzat la biblioteca gràfica swing, ja que és la que està per defecte amb el compilador NetBéans.

Una de les altres característiques del Java que he utilitzat és el multifils, ja que vaig comprovar que a l'hora de llegir el port sèrie el programa es quedava a l'espera de que l'acció acabés, per tant vaig utilitzar aquesta possibilitat per poder continuar utilitzant el programa.

Capítol 7

Metodologia

Per realitzar aquest projecte s'ha considerat una divisió de fases per poder garantir el seguiment i la consecució del mateix. El projecte està format per quatre fases:

- *FASE 1 – Disseny Físic i de la placa base.*

En aquesta primera fase, s'ha realitzat el disseny físic del robot. En un primer moment el material elegit va ser l'alumini, pel fet de ser més lleuger però el seu preu em va fer tirar endarrere. Després de molt pensar vaig acabar elegint fusta, però fusta de contraxapat de xop, ja que també és un material lleuger i més barat que l'alumini. Un cop elegit el material, va tocar realitzar el disseny. El disseny és senzill, intentant mantenir el pes màxim a la part baixa del robot, amb una altura que permeti a una persona d'altura estàndard col·locar els documents de manera còmoda i una tapa que ens deixi accedir a l'interior del robot també còmodament.

Un cop el robot ja muntat, toca realitzar el disseny de la placa PCB, mitjançant l'aplicació Eagle 5.11.0.

- *FASE 2 – Programació Atmega32u4*

Un cop muntat el robot, es passarà a la programació del microcontrolador amb la utilitat WinAVR i Flip 3.4.5. En aquesta fase escriuré les funcions d'inicialització de cada component i la seva funció, així com les interrupcions necessàries per realitzar els pertinents càlculs.

- *FASE 3 – Modelat de la part elèctrica*

La següent fase és realitzar la part elèctrica. Aquesta és una fase molt problemàtica, ja que com explicaré més endavant, he separat la bateria que donarà corrent a la part electrònica i la de la part de potència (Ponts en H i motors). En ella trobem un interruptor doble ja que en el cas d'estar el robot apagat, les bateries es col·loquen en paral·lel, això fa que quan estigui endollat a la corrent externa es carregaran les dues bateries. També en el disseny he pensat que seria important col·locar un fusible en la part de potència, ja que és la part més delicada.

- *FASE 4 – Control del robot i proves finals*

Per aquesta fase he pensat que el millor és realitzar una aplicació JAVA on, gràcies al mòdul Buetooth que he integrat al robot, podrem variar algunes variables i veure com es comporta el robot , així com obtenir les dades dels sensors i actuadors.

Posteriorment, també s'ha integrat en aquesta aplicació un control remot via Buetooth.

Un cop realitzada l'aplicació procedirem a l'ajust del robot i les diferents proves en el prototip real.

Capítol 8

Arquitectura i planificació

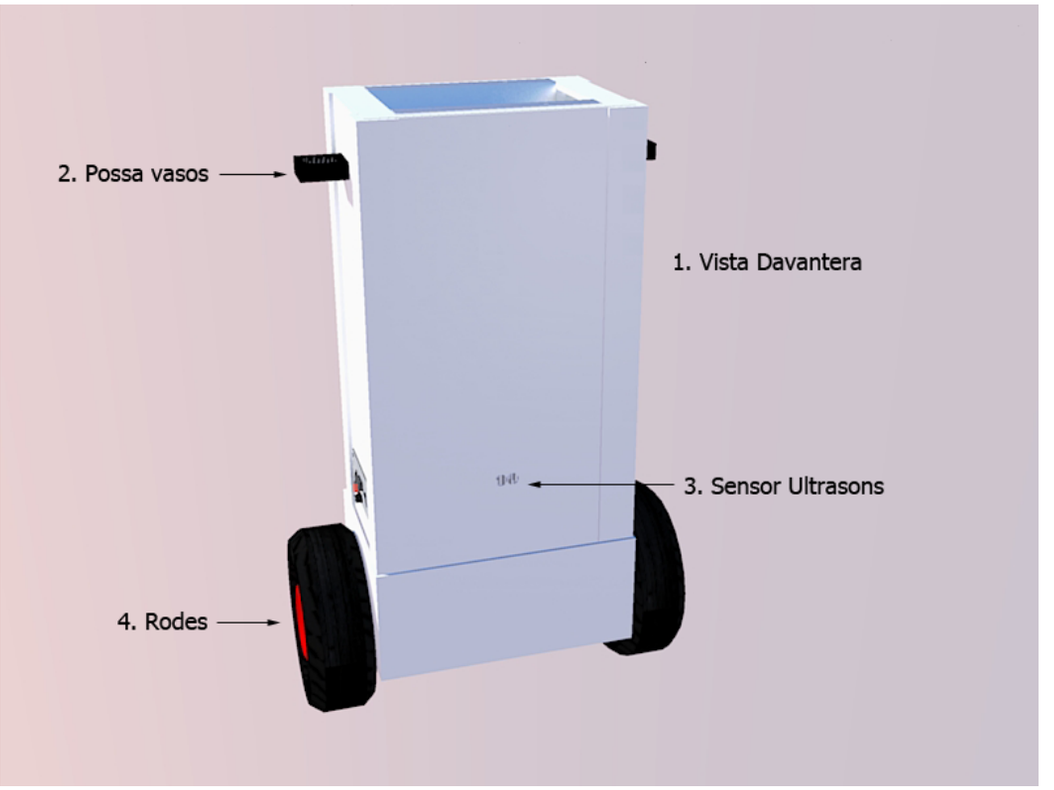


Figura 8.0.1: Vista frontal 3D

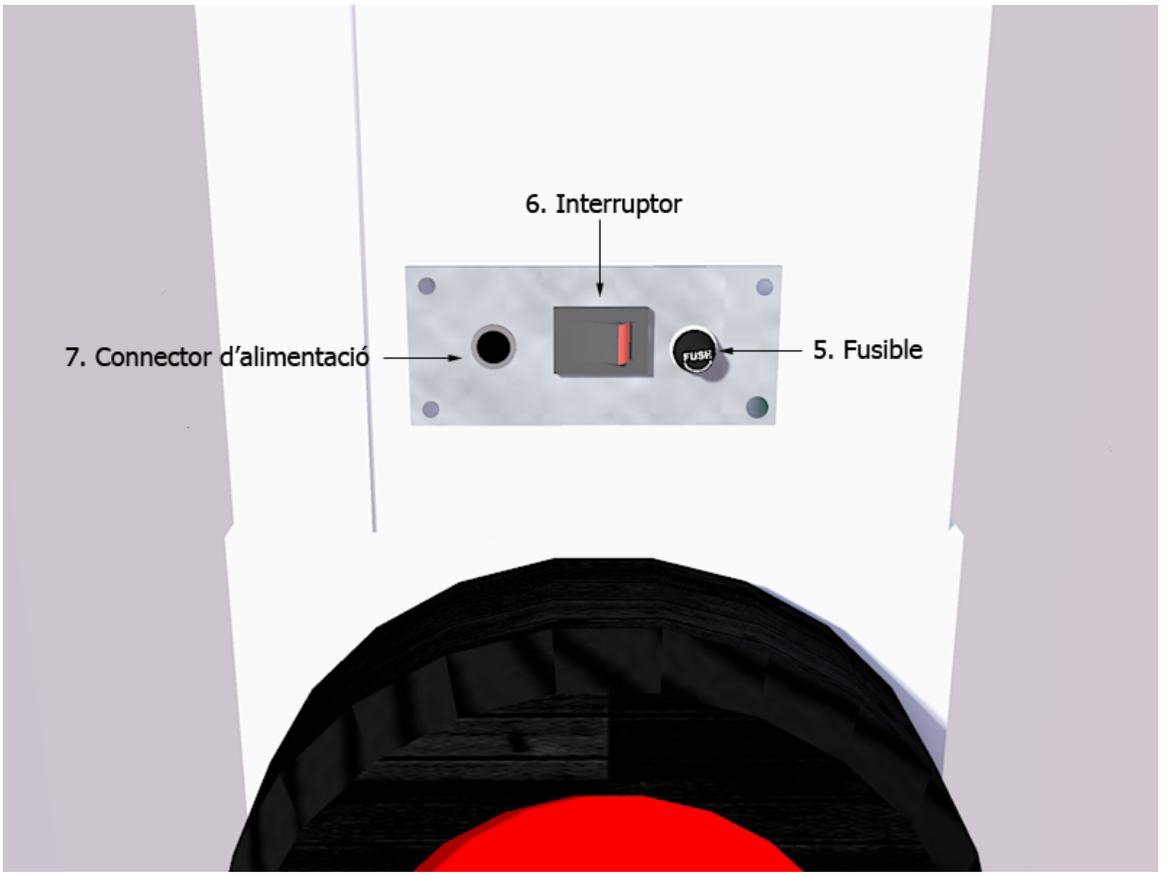


Figura 8.0.2: Vista lateral 3D

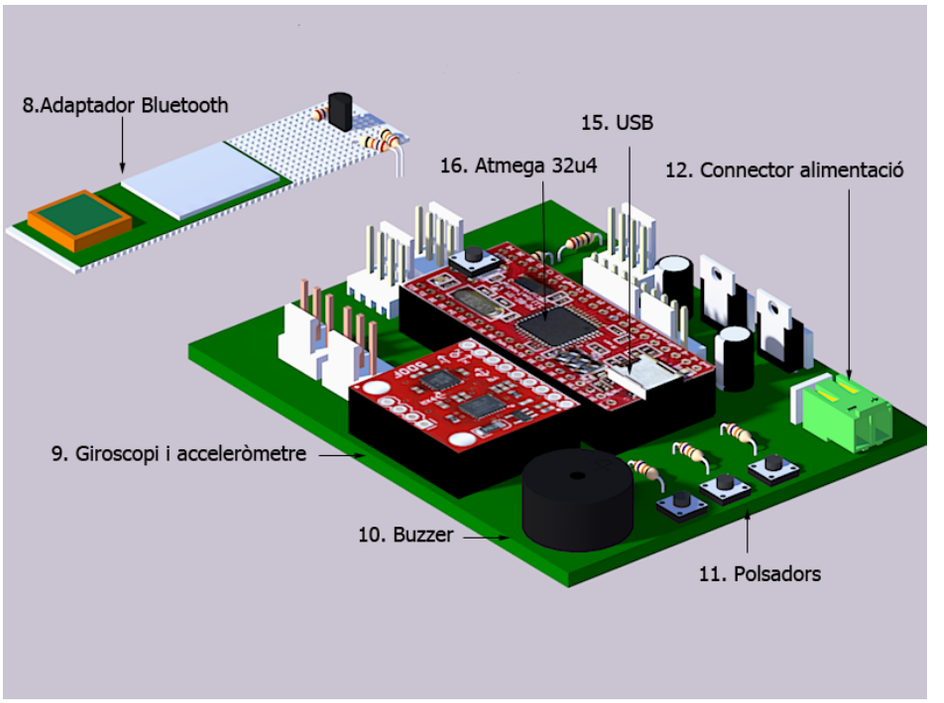


Figura 8.0.3: Vista placa 3D

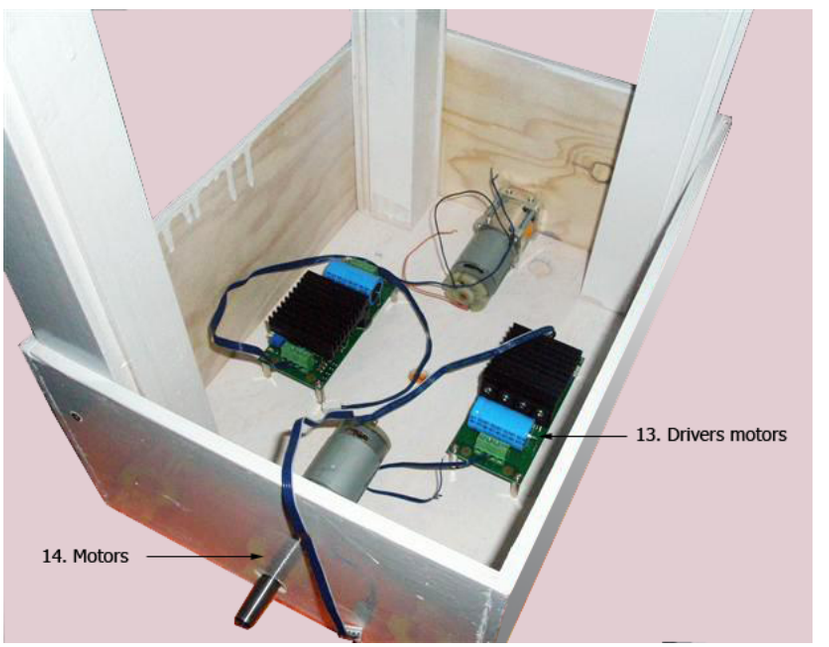


Figura 8.0.4: Vista interior 3D

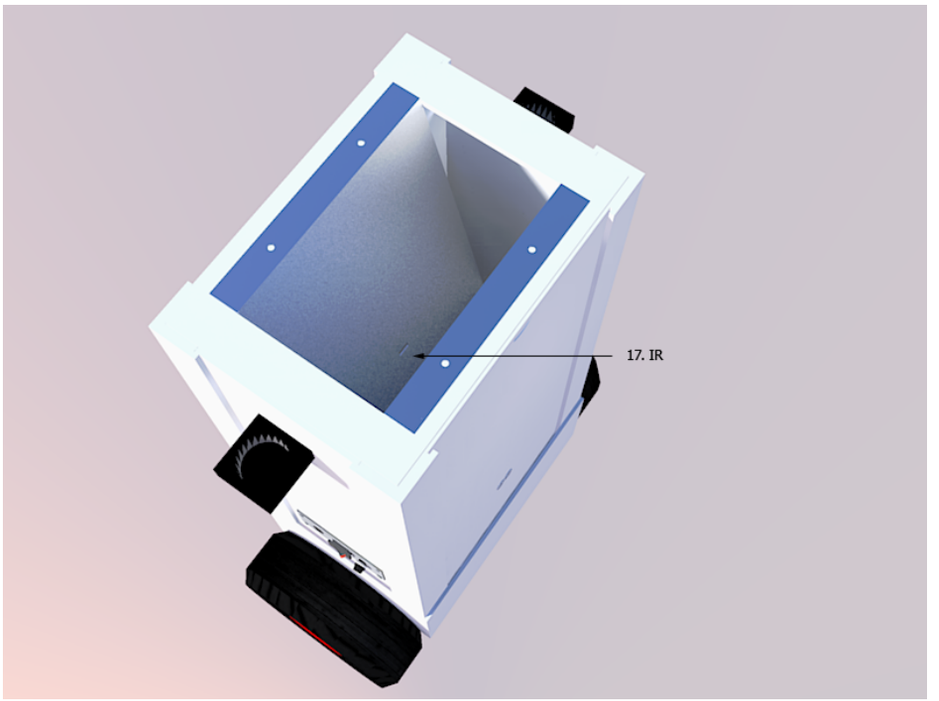


Figura 8.0.5: Vista sobre 3D

Número	Nom	Descripció
1	Vista Davantera	Part davantera del robot
2	Posa Vasos	Accessori que ens permet transportar llaunes
3	Sensor Ultrasons	Sensor de distància
4	Rodes	Rodes de 22 polsades
5	Fusible	Fusible de 10A
6	Interruptor	Interruptor on/off
7	Connector d'alimentació	Connector per carregar les bateries de 12V
8	Adaptador Bluetooth	Adaptador per la comunicació sèrie amb el pc
9	Giroscopi i acceleròmetre	Per calcular l'angle i la velocitat angular
10	Buzzer	Alarma sonora quan troba un obstacle
11	Polsador	Polsadors de proves
12	Connector alimentació	Connector 12V per la placa
13	Drivers motors	Control dels motors
14	Motors	Motors 900rpm 26:1 Gearmotors
15	USB	Port de comunicació sèrie
16	Atmega32u4	Microcontrolador principal
17	IR	Sensor detector de documents

Taula 8.1: Components

A continuació es presenta la planificació temporal del projecte i el pressupost (8.2) dels materials empleats.

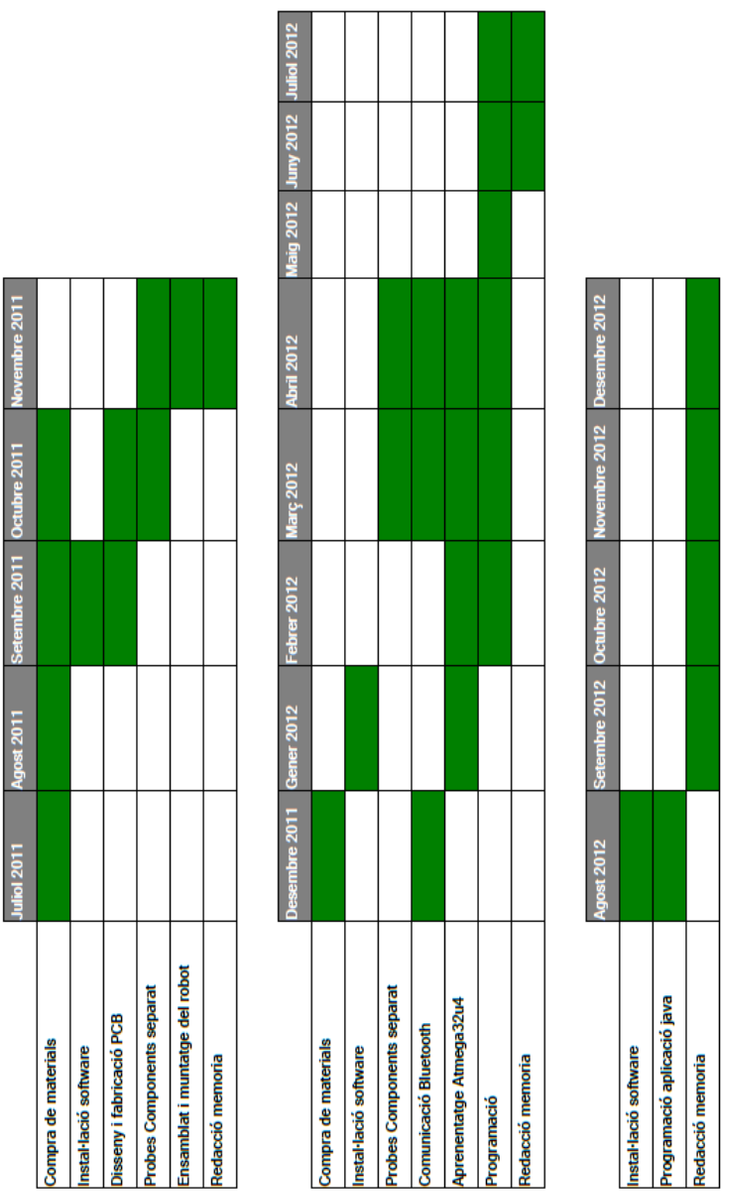
Aquestes són algunes de les botigues on he comprat material [PRE12, ROB12].

Materials	**Preu**
Atmega 32u4	15,86€
24V 20A H Bridge Speedo	118,74€
QTR-8RC Reflectance Sensor	12,41€
Pololu Distance Sensor Sharp GP2Y0D810Z0F	5,77€
SRF05 Ultrasonic Range Finder	13,56€
Shipping cost	11,26€
PCB	15€
IMU Analog Combo Board - 5 Degrees IDG500/ADXL335	34,86€
IMU Analog Combo Board - 5 Degrees IDG500/ADXL335	34,86€
Components	20€
Estructura	0€
PDX26 - 26:1 Gearmotor	148,6€
Peces alumini	80€
Llicencia Eagle Hobbyist	140€
Mà d'obra	3500€
Rodes	10€
Vinil	15€
Total	**4175,92€**

Taula 8.2: Presupost

	Juliol a Novembre 2011	**Desembre 2011 a Juliol 2012**	**Agost a Desembre 2012**
Fase 1 - Compra materials, instal·lació software, disseny i fabricació robot i probes inicials	☑		
Fase 2 - Probes, aprenentatge i programació robot		☑	
Fase 3 - Instal·lació software, programació aplicació java i redacció memoria			☑

Taula 8.3: Planificació temporal - Visió general



Capítol 9

Disseny

9.1. Disseny Físic del robot mitjançant Google SketchUp

En aquesta fase es va dissenyar el robot utilitzant l'aplicació Google SketchUp 8. Podem veure la seva interfase gràfica en l'apartat (Veure programés). Gràcies a aquest programa podrem determinar si les mides calculades de cada peça podran formar el robot correctament.

La idea principal era agafar com exemple el robot Segway, però al no ésser controlat per una persona va fer que em decidís per fer un disseny menys atractiu i completament sòlid, cosa que em va facilitar el col·locar els components interns d'una manera més fixa. Aquest fet va fer que s'hagués de dissenyar totes les peces que formen part del muntatge del robot. El primer que s'havia de decidir era el material que formaria l'estructura externa. La primera idea era formar el robot amb peces d'alumini, però el preu de l'alumini va fer que no em decidís per aquest material. Al final vaig decidir fer l'estructura de fusta, ja que el preu era molt més atractiu que l'alumini i era molt més fàcil de modelar.

Com a conseqüència he dissenyat 11 peces per aquest projecte.

9.1.1. Peça 1 - Superior

Peça superior (9.1.1), és aguantada per les 4 peces principals i on col·locarem la 'V' per col·locar els documents.

9.1.2. Peça 2 - Laterals

Quatre peces (9.1.2) com aquesta són les més importants del robot, són com les bigues que aguanten tota l'estructura.

9.1.3. Peça 3 - Caixa

És on col·locarem la peça 2 a les quatre cantonades, d'aquesta peça (9.1.3) només en tenim una i és on tenim els motors i els drivers dels motors, ja que és la més estable i més robusta. Així doncs la força exercida pels motors no afectarà a l'estructura ni als sensors principals.

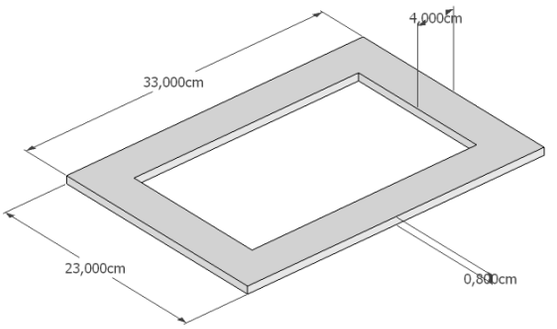


Figura 9.1.1: Vista peça 1

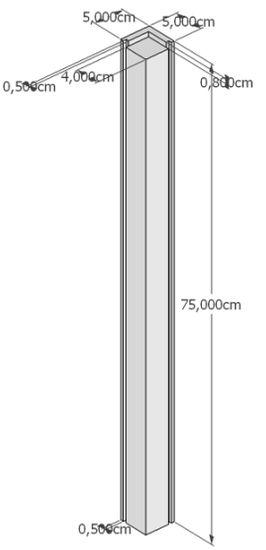


Figura 9.1.2: Vista peça 2

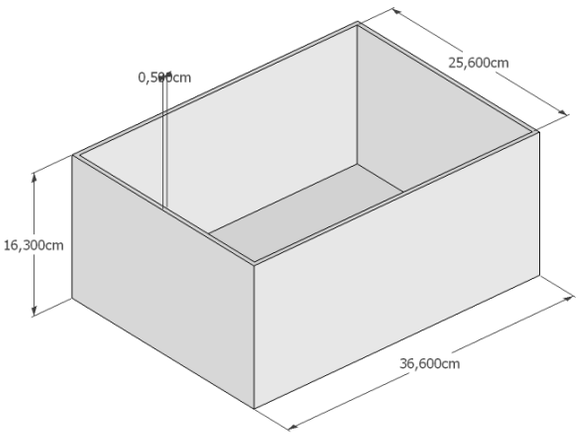


Figura 9.1.3: Vista peça 3

9.1.4. Peça 4 - Tapa posterior

Tapa posterior del robot (9.1.4), el seu ús principal és tapar la part de darrera del robot. Veurem que és diferent a la tapa davantera només en el seu gruix, ja que, aquesta tapa és fixa i s'ha d'adaptar perfectament a les guies de la peça 2.

9.1.5. Peça 5 - Tapa davantera

Tapa davantera del robot (9.1.5). A diferència de la peça 4, aquesta peça no és fixa ja que ha de lliscar per les guies de la peça 2 sense exercir cap tipus de força. El fet de no fer-la fixa ajuda a que podrem accedir a l'interior del robot per fer qualsevol canvi, com per exemple, de bateries...

9.1.6. Peça 6 - Laterals

Peça lateral del robot. D'aquestes peces (9.1.6) en tenim dos, una per cada lateral del robot. A l'igual que la peça 4 serà fixa, ja que no volem accedir a l'interior del robot pels laterals. Es diferència de la peça davantera i posterior per la seva amplada, ja que sinó el robot seria massa gros.

9.1.7. Peça 7 - Interior

Aquesta peça és interior (9.1.7), estarà col·locada al centre del robot, adaptada per unes petites 'T' que també estaran enganxades a la peça 2. Aquesta peça és molt important, ja que en ella adaptarem la Placa amb el microcontrolador, el Bluetooth i els sensors més importants, el giroscopi i l'acceleròmetre. També contindrà les bateries.

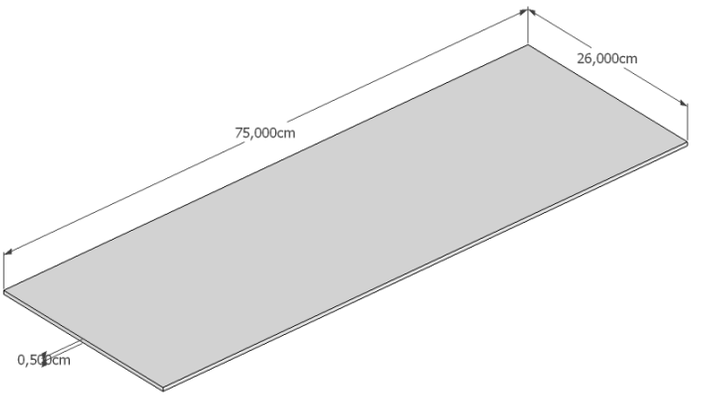


Figura 9.1.4: Vista peça 4

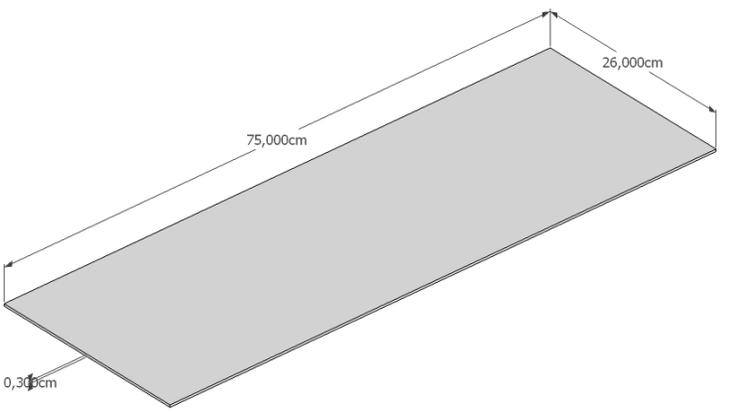


Figura 9.1.5: Vista peça 5

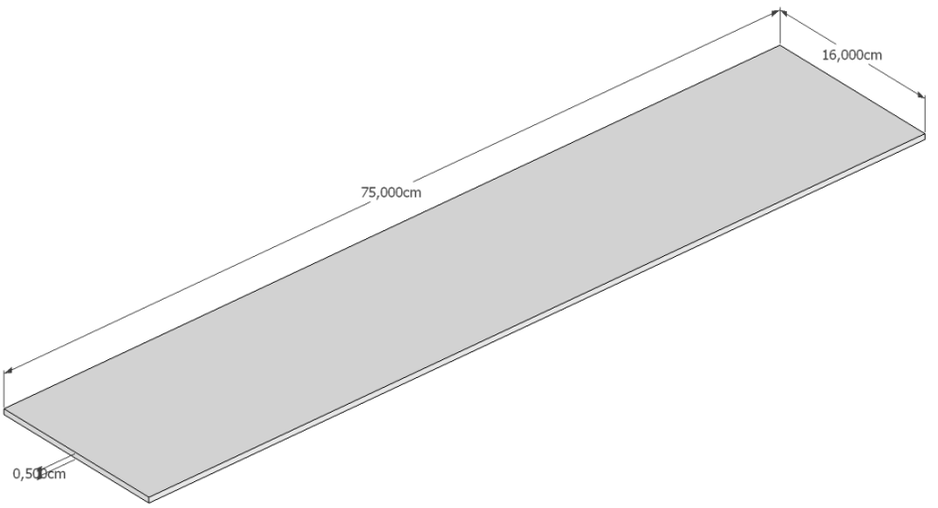


Figura 9.1.6: Vista peça 6

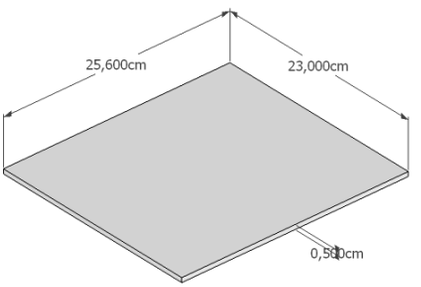


Figura 9.1.7: Vista peça 7

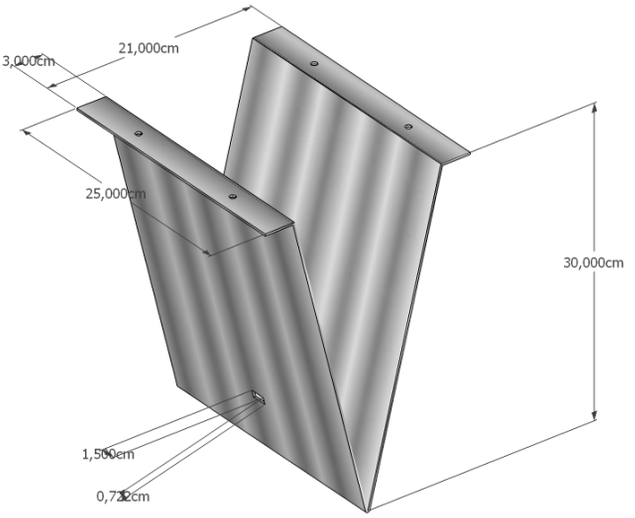


Figura 9.1.8: Vista peça 8

9.1.8. Peça 8 -'V'

'V' per als documents. Aquesta és la única peça (9.1.8) feta amb alumini, ja que era la més complicada de fer i era necessari que fos adaptable per poder entrar perfectament en la peça 1. Podem observar en un lateral un forat: en ell col·locarem el sensor d'infrarojos, així detectarem si hi ha algun document o no.

9.1.9. Peça 9, 10, 11 - Caixa inferior

Aquestes 3 peces (9.1.9) formaven la caixa on tindrem els sensors d'infrarojos per utilitzar-los per seguir una línia. Estava col·locada sota la peça 3 i les mides eren perfectes per aguantar qualsevol moviment sense tocar a terra. Al final no s'utilitzarà ja que era gairebé impossible que un robot equilibrista seguís una línia sense sensors encoders. Més endavant explicaré el perquè de la necessitat d'aquests sensors.

En la figura 9.1.10 veiem la forma final de les peces 9, 10 i 11 perfectament ensamblades.

Una vegada en disposició de totes les peces, es procedeix al ensamblat del robot. Per ell, s'ha dividit la construcció en diferents etapes. De forma que qualsevol modificació en alguna de les etapes s'heretarà automàticament en totes aquelles que la utilitzin. L'ensamblat també s'ha fet de manera asíncrona.

9.1.10. Vistes

En les següents figures mostrem les diferents vistes del model del robot en Google SketchUp.

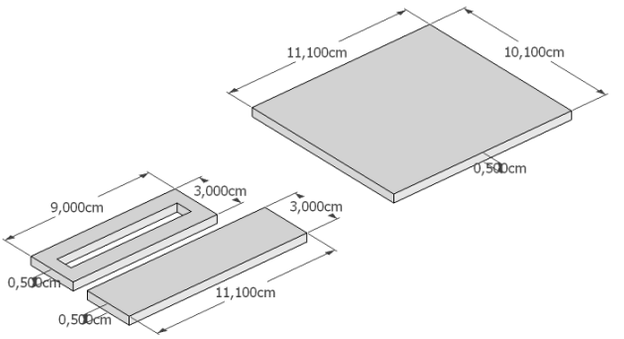


Figura 9.1.9: Vista peça 9, 10, 11

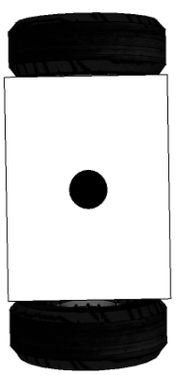


Figura 9.1.11: Sota

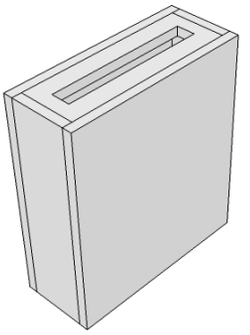


Figura 9.1.10: Vista final peça 9, 10, 11

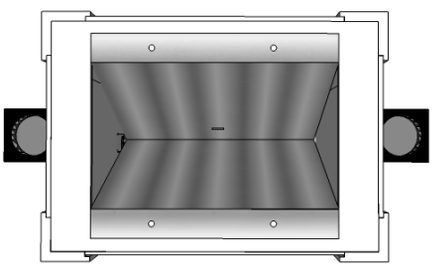


Figura 9.1.12: Sobre

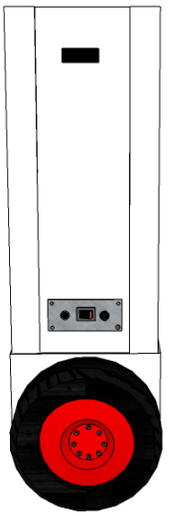


Figura 9.1.13: Costat

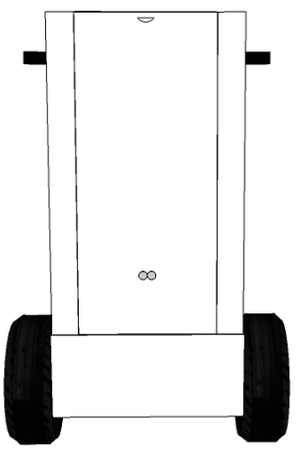


Figura 9.1.14: Davant

9.2. Disseny PCB mitjançant Eagle

Com observem en la figura 9.2.1, el mòdul de control interactua amb tots els mòduls del robot. El mòdul de control és la part fonamental del robot, convertint-se en un dispositiu capaç de rebre, processar i generar senyals a partir de la informació que l'hi arriba des dels altres mòduls. En les

següents figures veiem detalladament com està estructurada la placa que conté aquest mòdul de control.

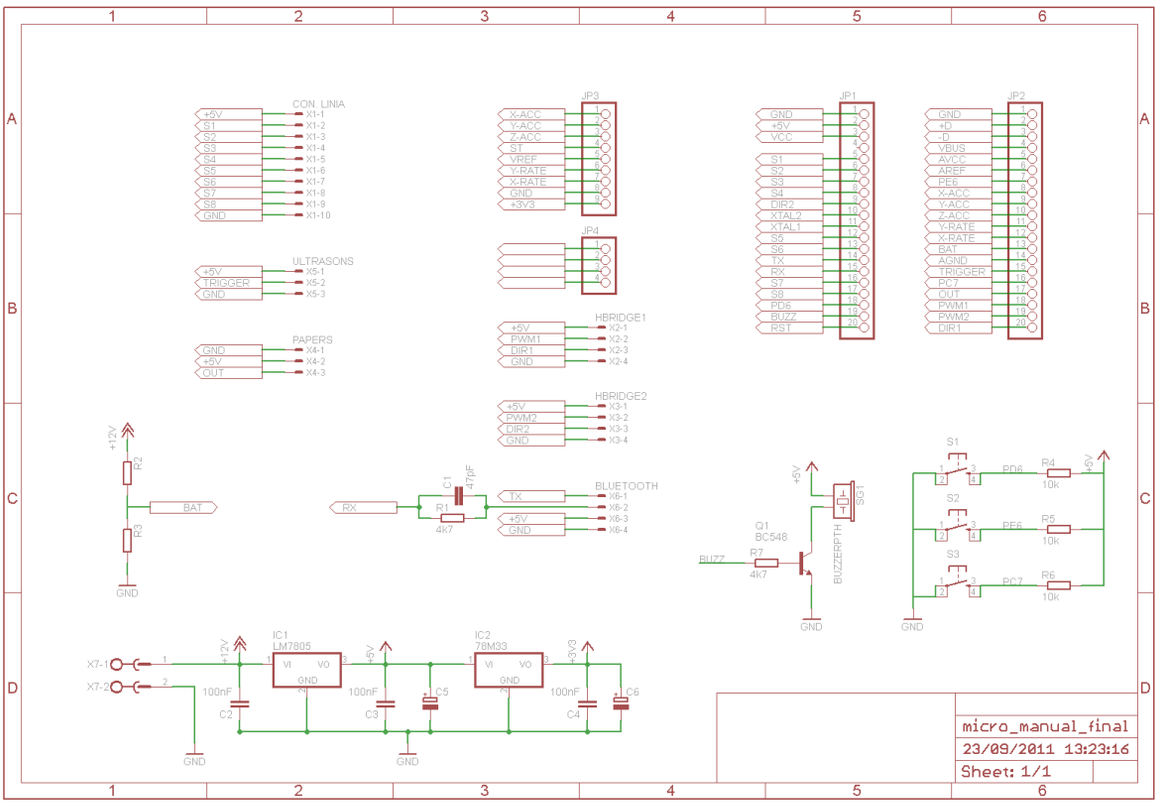


Figura 9.2.1: Esquema PCB

Tenint tots els components del disseny de l'electrònica de control dimensionats, es crea un esquema teòric amb els dissenys implementats per l'aplicació en concret i després un esquema (9.2.2) de la placa física amb el software de disseny Eagle. Aquesta placa és de doble cara, però només té components en la part de sobre, en canvi té pistes en les dos cares. Els components són de tipus convencionals i per tant es solden travessant la placa.

La PCB es realitza per insolació, on primer creem la placa de coure i després a partir d'un líquid àcid ens mengem el coure sobrant. Veiem el resultat final en la figura 9.2.3

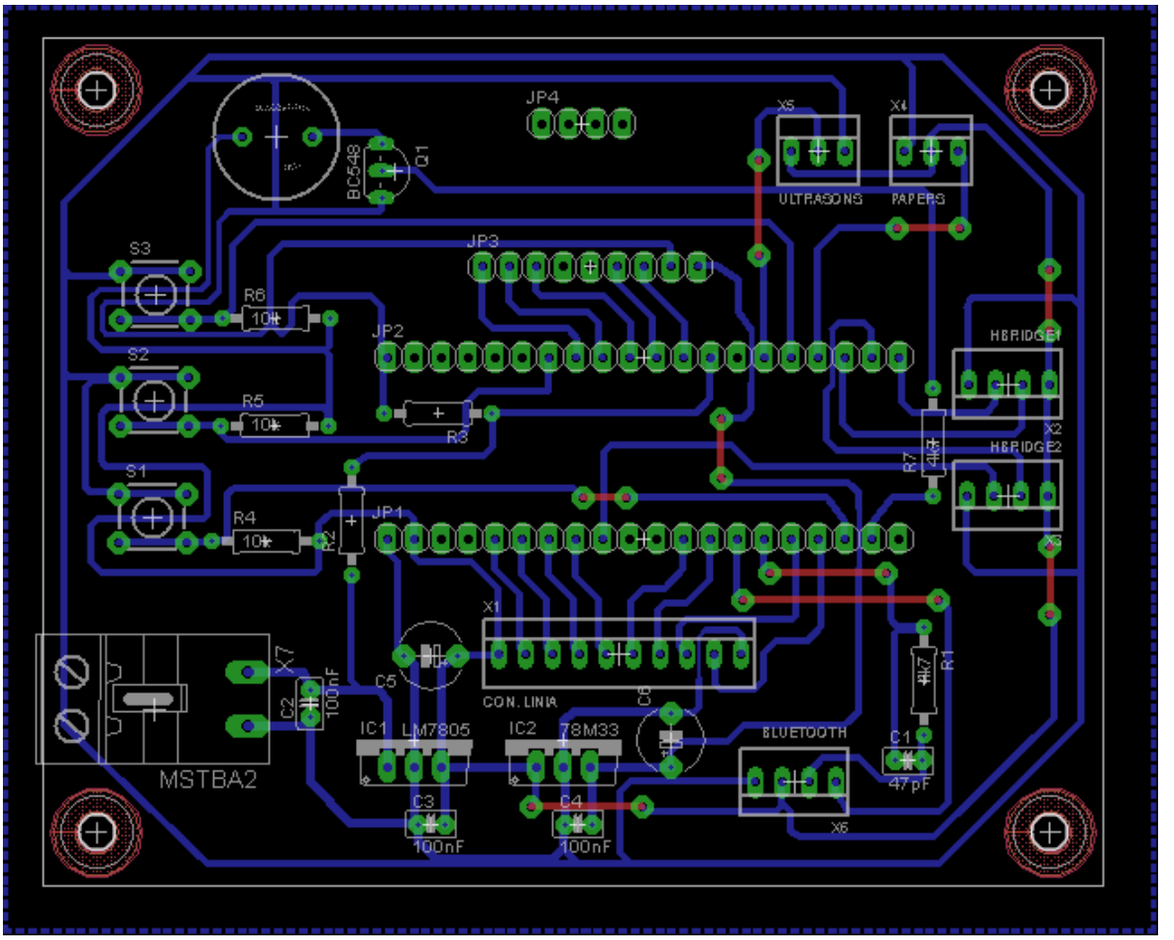


Figura 9.2.2: Board robot

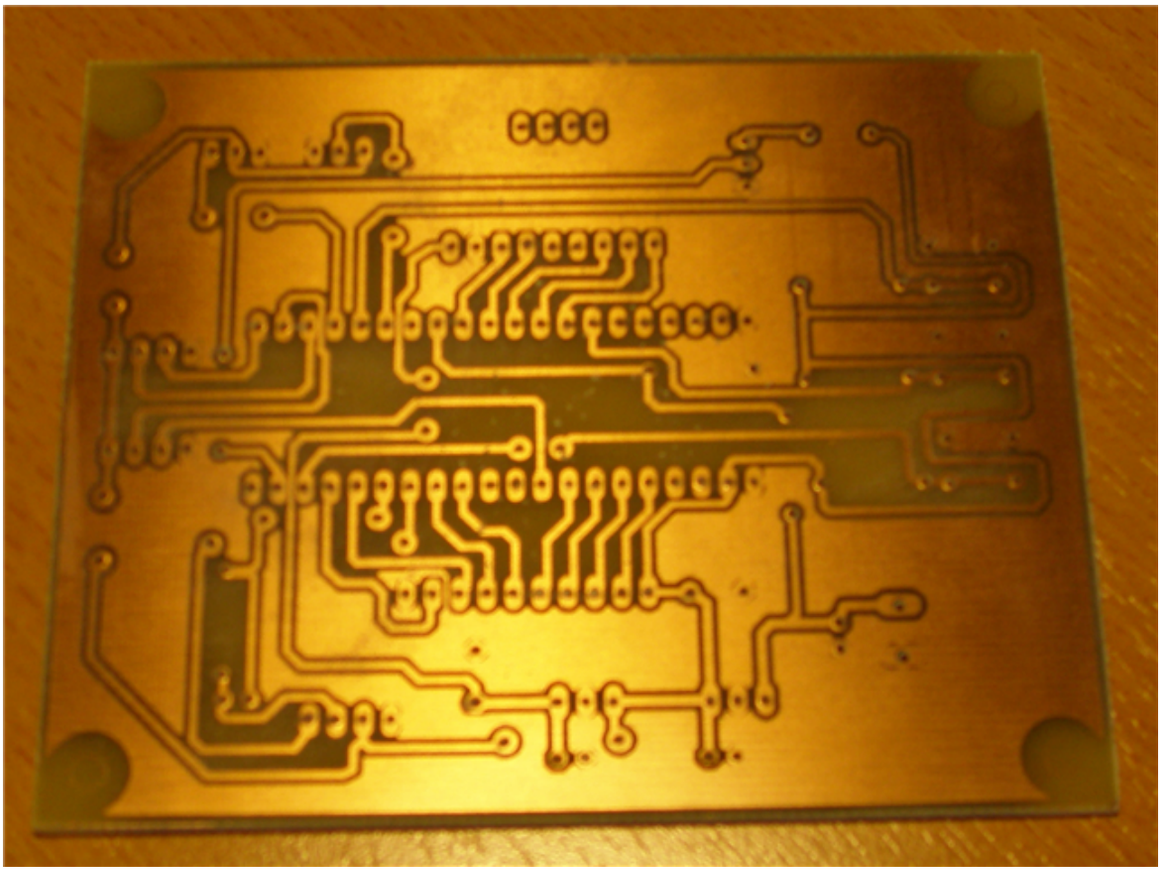


Figura 9.2.3: Placa final

9.3. Disseny elèctric

La funció principal de la part de potència és garantir el correcte abastiment d'energia de cada un dels mòduls del robot. Per això es requereix un disseny robust que permeti donar el voltatge necessari a tots els mòduls.

En la figura 9.3.1 representem de manera gràfica com està organitzada la part de potència.

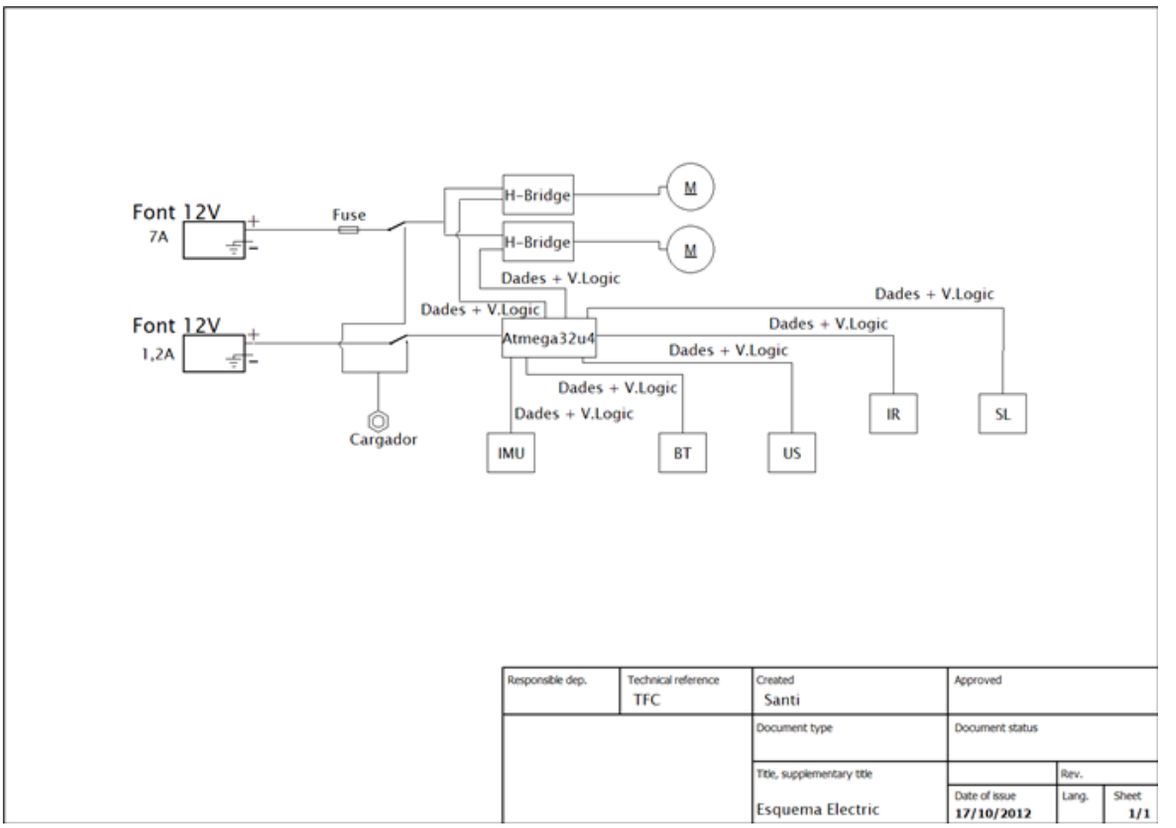


Figura 9.3.1: Esquema disseny elèctric

Capítol 10

Components

Un cop tenim totes l'estructura física, la placa i l'esquema elèctric definit veurem els components necessaris perquè el funcionament del robot sigui el desitjat. El preu ha estat la raó principal per haver triat la majoria de components, menys dels motors i dels drivers del motor, ja que com actuadors són els components més importants per mantenir en equilibri el robot. El problema principal sempre ha estat l'espera, ja que si volia components bons i barats s'havia de comprar principalment a Xina i Estats Units, cosa que va provocar que els temps d'enviament fossin alts i això sense tenir en compte alguns errors en l'enviament per part de la tenda en qüestió.

Seguidament veurem els components elegits pel projecte:

- **Distance Measuring Sensor Unit -** Mesurador de distàncies per IR

El primer component és un mesurador de distàncies via infrarojos. Compost d'una combinació integrada de PD i IRED i circuit de processament de senyal. La tensió de sortida d'aquest sensor es manté alta en el cas que hi hagi un objecte en el rang de distància específica.

Vaig elegir aquest sensor pel fet que portava el circuit de processament de senyal, ja que facilitava molt la feina a l'hora de programar el microcontrolador. Aquest sensor és l'encarregat de comprovar si hi ha un document en la "V" del robot, per tant la distància màxima de detecció de 100mm és més que suficient.

En la imatge 10.0.1 veiem com és físicament el mesurador de distàncies.

Veure datasheet en l'Annex, a la pàgina 119.

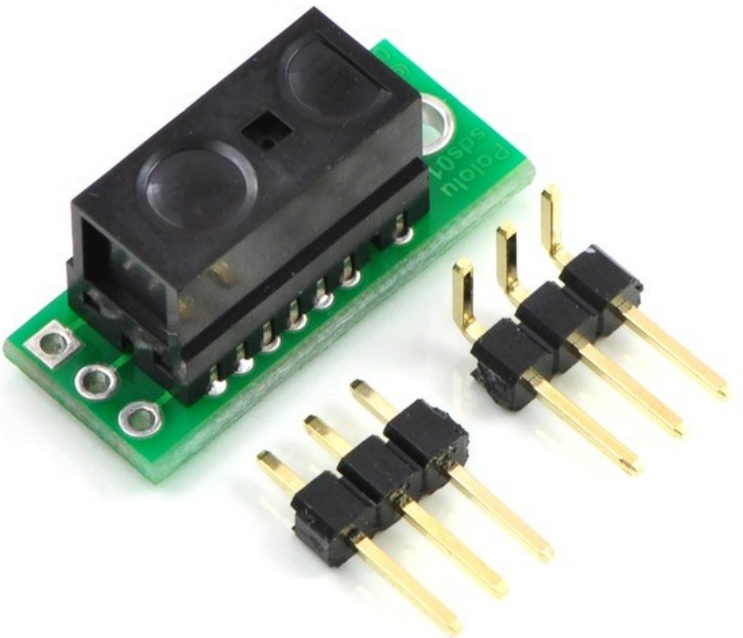


Figura 10.0.1: Distance Measuring Sensor Unit

- **Ultra-Sónic Ranger SRF05 -** Mesurador de distàncies per Ultrasons

El sensor d'ultrasons SRF05 (10.0.2) es tracta d'un mesurador de distàncies de baix cost. La detecció de l'objecte consisteix en mesurar el temps que tarda en rebotar un feix d'ultrasons en una superfície, aquest fet em va fer decidir per aquest tipus de sensor per poder detectar obstacles.

Internament, aquest sensor està constituït per un microcontrolador i dos càpsules ultrasòniques de 40khz. Una per disparar i l'altra per rebre.

Hi ha tres característiques principals que em van fer decidir per aquest sensor, la primera és el rang de mesura màxima, que és de 4 metres. La segona és el mode de 3 fils, dels quals només 1 és de dades, cosa que va fer que amb una sola entrada/sortida del microcontrolador pogués utilitzar aquest sensor. I la última i no la menys important, el preu.

Veure datasheet en l'Annex, a la pàgina 119.

Figura 10.0.2: Ultra-Sónic Ranger SRF05

- **Intélligent Bluetooth™ Serial Mòdule BISMS02BI-01** - Adaptador Bluetooth

Per aquest projecte he elegit el mòdul de comunicació Bluetooth BISMS02BI-01 (10.0.3) de la marca Ezurio amb una cobertura assegurada en pla de 100m (classe 1), aquest mòdul ens assegura un bon radi de treball sobre el robot, per exemple en empreses on hi hagi més d'una planta. Una altra de les característiques importants és que pot comunicar amb un microcontrolador a través de port sèrie UART (Universal Asinchronous Receiver Transmittér), que és l'estàndard RS-232, amb el que podem tractar les dades rebudes i també enviades. Una altra característica important és que incorpora integrada una antena en el mateix dispositiu. Realitza les transmissions inal·làmbriques per radiofreqüència en el rang 2,400 i 2,485 Ghz i l'energia de transmissió màxima a la que arriba és de +6dBm i la mínima -27dBm, cosa que fa que el seu consum sigui només de 36mA amb funcionament normal de transmissió. El mòdul disposa de 6 pins E/S i la possibilitat de configurar un ADC de fins a 8 bits, però amb el problema que necessita un processador addicional si volem processar les dades que agafaria el conversor. Aquesta opció no és utilitzada en aquest projecte, per això no es una característica rellevant. La configuració de l'integrat es realitza a través de port sèrie amb comandes estàndards AT.

Veure datasheet en l'Annex, a la pàgina 119.

Figura 10.0.3: Intélligent Bluetooth™ Serial Mòdule BISMS02BI-01

- **Mòdul MD03** - Control dels motors

El MD03 (10.0.4) és un controlador per motor de corrent contínua de mitjana potència, dissenyat per proporcionar més potència que els controladors basats en un únic circuit integrat. Les principals característiques són la facilitat d'ús i la flexibilitat. La potència del motor és controlada mitjançant el PWM del pont H a una freqüència de 7,8Khz.

El MD03 està preparat per aguantar molta corrent, però és necessari tenir algunes precaucions a l'hora de fer el cablejat. És molt important evitar que la corrent del motor retorni al circuit lògic. Així que l'electrònica i el motor haurien d'estar connectats a bateries diferents. Això em va donar molts problemes ja que no ho vaig tenir en compte, la primera vegada en el muntatge i al posar-ho en funcionament, que el motor es parava, cosa que va fer que la búsqueda del problema fos un autèntic maldecap. El fet que es pares el motor era degut a que utilitzava una sola bateria per alimentar la part electrònica i de motor i això feia que el xip del driver es sobrealimentés i acabés cremant-se. Al final, el fet de connectar les dues parts amb bateries diferents va solucionar el problema.

Un altre canvi que vaig fer en el mòdul va ser treure el microcontrolador que portava, un PIC, ja que volia que fos el meu controlador Atmega32u4 el que controlés la senyal PWM.

La característica principal que em va fer decidir per aquest Pont H va ser la seu preu, ja que no tenia un preu excessivament alt per la qualitat que tenia. Amb un dissipador i 4 MOSFET's era ideal per

Figura 10.0.4: Mòdul MD03

aguantar fins a 20A, cosa que feia que pogués arribar al màxim d'ells sense necessitat de patir per un sobreescalfament.

Veure datasheet en l'Annex, a la pàgina 119.

- **QTR-8RC** - Sensors IR per seguir línies

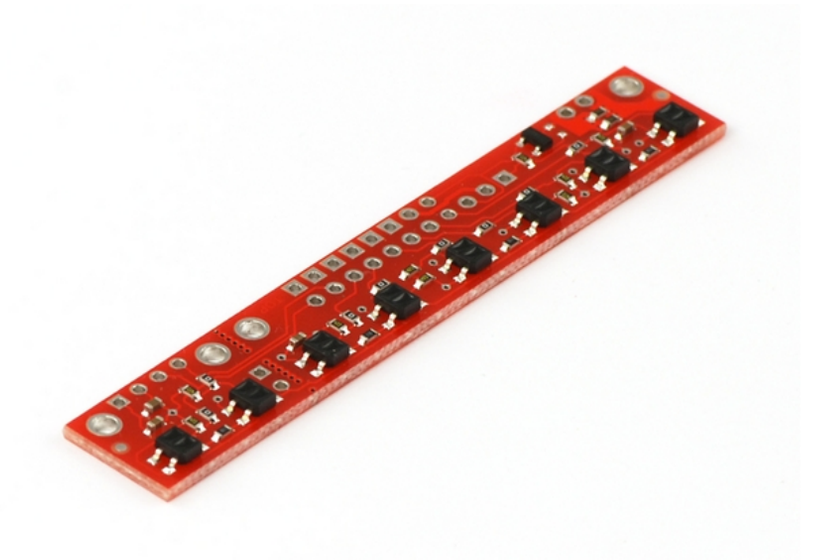


Figura 10.0.5: QTR-8RC

El QTR-8RC (10.0.5)([QTR11]) és un sensor de reflexió IR, aquest dispositiu electrònic incorpora un emissor IR i un fototransmissor en el mateix encapsulat. Com que l'emissor i el receptor estan en el mateix pla, el funcionament consisteix el que el díode emissor emet una llum infraroja i la llum que és reflexada en la superfície és detectada pel fototransmissor.

Aquest sensor es caracteritza per llegir la línia mitjançant el temps de carga d'un condensador a través del fototransistor. Aquesta característica va ser el que em va fer adonar que m'havia equivocat decidint-me per aquest sensor, ja que hi havia sensors que podien llegir directament en digital, 1 o 0, i que realment ho fa més senzill a l'hora de programar. Vaig prendre la decisió de comprar aquest sensor pel fet que els 8 sensors que porta em feia pensar en alguns algoritmes per seguir les línies i que creia que podrien anar força bé.

Tot i això vaig utilitzar-lo. Les proves amb una cinta aïllant negra sobre una superfície blanca van ser bastant bones, tot i que quan va ser ensamblat amb el robot la cosa va canviar. El fet que el robot mantingués l'equilibri amb només dues rodes provocava que el fet de seguir una línia era una missió massa complicada per ell, ja que la interrupció utilitzada per mantenir l'equilibri sempre era més important que el seguir la línia, cosa que feia que sortís de la seva trajectòria.

Veure datasheet en l'Annex, a la pàgina 119.

- **PDX26** - Motor

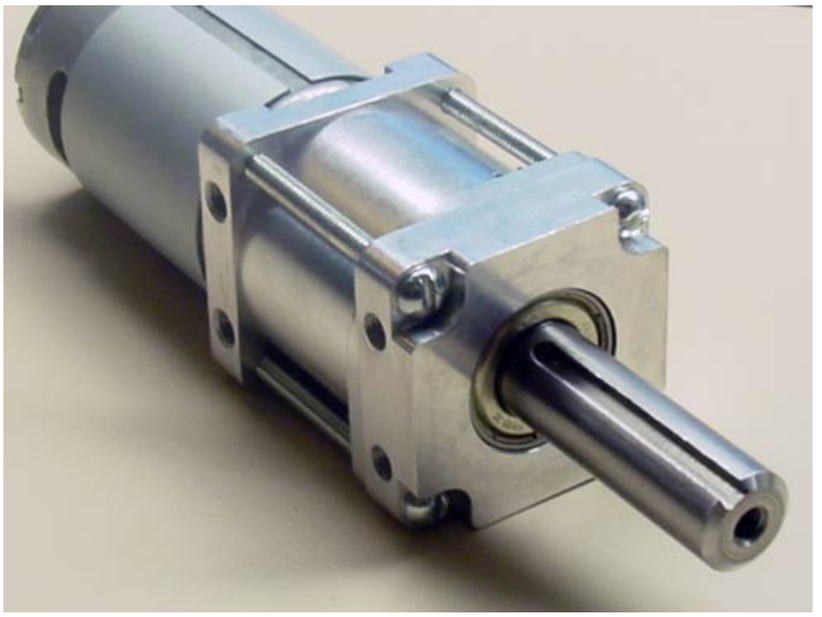


Figura 10.0.6: PDX26

Els motors utilitzats en el robot són 2 motors de corrent contínua amb la reducció que proporciona 900rpm sense càrrega amb un consum de 1,5A. L'eix del motor és una característica que no era rellevant, ja que vaig decidir fer una peça a mida per poder adaptar l'eix a les rodes. A més, la força que especifica el fabricant és de 16,38 N/m per motor.

Els motors van ser un dels components més difícils de triar, normalment un motor amb reductora amb unes condicions bones era molt car. Així que ja sabia que el preu dels motors seria alt. Així que vaig basar la meva decisió amb el fet que la velocitat, la força del motor i que el consum no fos massa alt. En la següent taula veiem una comparativa de 3 motors interessants.

Motor	Tensió (V)	Revolucions (rpm)	Força (N/m)	Consum sense carrega (A)	Preu (€)
Magnum 775 Planetary	12 - 14,4	720	17,6	3,8	69,9 .p
PDX26	12	900	16,38	1,5	66,04
PDX16	12	1500	10,28	1,5	62,1

Taula 10.1: Comparativa Motors

El criteri dominant en aquest cas és obtenir la màxima força possible sacrificant velocitat i obtenint el consum més baix possible, pel que la opció elegida és el motor de 900rpm i 16,38 N/m PDX26 (10.0.6).

He observat que un cop acoplada la roda ,el consum dels motors és una mica superior al nominal, però com els Ponts H poden aguantar fins a 20A no és un problema, només que amb el consum màxim

dels motors la bateria pateix una mica. Tot i que mentre no es doni una empenta al robot aquest consum es mantindrà estable.

Respecte a la seva col·locació, els motors es clavaran de tal manera que quedarà bloquejat el seu moviment en totes direccions, inclòs el gir sobre el seu propi eix.

En resum, els motors utilitzats tenen un preu alt però les seves prestacions pel que fa a consum i a la força elevada, a l'igual que la facilitat del muntatge evitant mecanismes, accessoris i les seves conseqüències (desgast dels engranatges, augment de pes, número de peces mòbils excessives, etc.) fan que sigui la opció més adequada per aquest robot.

- **MT-DB-U4 -** Placa desenvolupament Atmega32u4

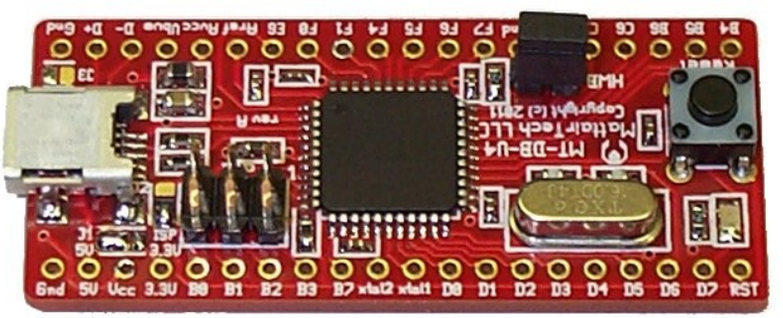


Figura 10.0.7: MT-DB-U4

La MT-DB-U4 (10.0.7)([MTD11]) és una placa de desenvolupament pel microcontrolador Atmel Atmega32u4. La placa està composta de 40 pins, i el seu muntatge és molt més fàcil que comprar un Atmega32u4 i soldar-lo directament a una placa. A més inclou un connector mini USB, un cristall de 16Mhz, un polsador de reinici i porta precarregat un bootloader DFU, cosa que és bastant interessant ja que no cal comprar un programador extern per carregar el nostre firmware.

La placa pot ser alimentada a través de USB a 5V o per alimentació externa entre 3V i 5,5V. Els pins de la placa són els mateixos que els de l'Atmega32u4.

Vaig prendre la decisió d'utilitzar un Atmel ja que volia provar d'utilitzar un microcontrolador diferent al PIC. Aquest últim ja l'havia utilitzat per aplicacions semblants. Va ser una bona decisió, ja que és un microcontrolador fàcil d'utilitzar i bastant amigable.

Un cop presa la decisió d'utilitzar el Atmel em vaig decidir pel model Atmega32u4 ja que tenia el mateix número de portés que jo volia utilitzar, a més, el model amb USB em va permetre comunicar-me amb ell per fer proves abans d'utilitzar el Buetooth, i com no, la facilitat per programar-lo era enorme.

A més, aquest microcontrolador disposa de 12 Canals ADC de 10 bits, 32KB de memòria FLASH, 4 timers (dels quals dos eren essencials, un pel PWM i l'altre per la interrupció principal) i també compatible amb el protocol UART, SPI i I2C.

Personalment penso que va ser una gran decisió la compra d'aquesta placa amb aquest microcontrolador, ja que va fer que m'estalvies molt temps i diners.

Veure datasheet en l'Annex, a la pàgina 119.

- **Bateries**

Figura 10.0.8: Bateries 12V

Com he comentat anteriorment que he utilitzat dos bateries, una per alimentar la part electrònica i l'altra la part de potència. La part electrònica és alimentada per una bateria de 12V 1,2A i la de potència per una bateria de 12V 7A.

Les bateries són de plom amb elèctrodes del tipus gel. El principal problema de les bateries es el pes, almenys per la bateria de 7A, bastant més pesant que la de 1,2A. Per tant vaig decidir col·locar-les el més centrades possible del robot, en un punt on no afectés al moviment i no fes ballar més del normal al robot a l'intentar posar-se en equilibri.

Les bateries vaig poder aconseguir-les bastant baratés, per això em vaig decidir per aquestes (10.0.8).

- **IMU Analog Combo Board – IDG500/ADXL335** - Giroscopi i acceleròmetre

El sensor 10.0.9 és un dispositiu mesurador d'inèrcia (IMU), incorpora un giroscopi IDG500 (Per saber més, mirar [GYR12]) de dos eixos i un acceleròmetre ADXL335 de 3 eixos (Per saber més, mirar [ADX12]).

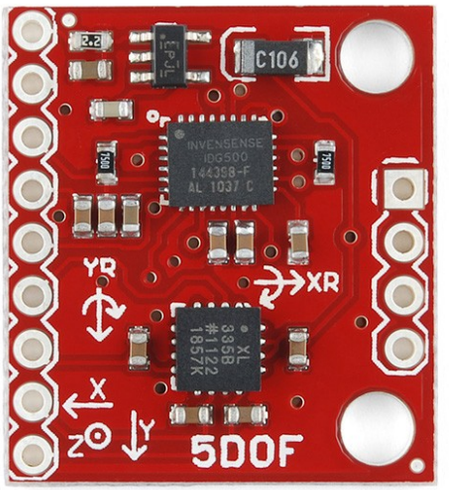


Figura 10.0.9: IMU Analog Combo Board – IDG500/ADXL335

Aquesta placa ens permet mesurar el Roll, Pitch, x, y z, en un espai menor de 2,5 centímetres i amb menys de 2 graus.

En primer lloc parlarem del acceleròmetre ADXL335. La acceleració és la rapidesa en què un cos guanya velocitat o la perd, podem mesurar aquesta acceleració en m/s2 o en forces G, que és la relació proporcional a la acceleració de la gravetat, és a dir, 1g són 9,8 m/s2. Les característiques que hem de tenir en compte a l'hora de decidir per un acceleròmetre són:

- Rang de mesures: Aquest rang defineix el límit superior i inferior que pot mesurar l'acceleròmetre. Aquest rangs com més petits són més sensibilitat tenim a la sortida. En el cas del ADXL335 és +-3.

- Intérface: La intérface es refereix a la manera com s'agafen les dades de sortida del acceleròmetre, normalment poden ser analògiques, digitals o PWM. En aquest cas el ADXL335 és analògic, cosa que el fa fàcil d'interpretar, ja que quan calculem una acceleració de 0g ens donarà la meitat del voltatge, i només tindrem que utilitzar un ADC per agafar la dada.

- Número d'eixos que mesura: De normal, un acceleròmetre pot mesurar 2 o 3 eixos. El ADXL335 és un acceleròmetre de 3 eixos, i pot mesurar x, y i z tot i que pel projecte només necessitaré 2 eixos.

Com podem observar el rang de mesures no és molt alt, però analitzant el meu projecte i veient que la velocitat de moviments no és molt alta, tampoc són necessaris uns límits elevats de mesures. Un cop vistes les característiques del acceleròmetre toca parlar del giroscopi.

Al contrari del acceleròmetre, al giroscopi no l'afecta la gravetat. Al giroscopi l'afecta la velocitat centrípeta del moviment de rotació, el qual quan està parat la tensió és 0. Aquesta velocitat es representa per rpm o graus per segon (^{o}/s).

Les característiques en què m'he fixat pel giroscopi són les següents:

- Màxim valor de mesura: Defineix el valor màxim de la velocitat angular que pot mesurar el giroscopi. En el cas del nostre giroscopi es de +-110^o/s, suficient pel nostre projecte.

Figura 10.0.10: Rodes 26 cm

- Intérface: A l'igual que l'acceleròmetre, les dades les rebrem de forma analògica i les convertirem amb un ADC del nostre microcontrolador.

- Número d'eixos que mesura: El més normal és que un giroscopi de gamma mitja només mesura dos eixos, el x i el y. En el nostre cas només utilitzarem l'eix x pel projecte.

Veient que alguns giroscopis mesuren fins a $6000^{o}/s$, la velocitat màxima del IDG500 no és molt alta, però considerarem que el robot no passarà de $30^{o}/s$ ja que en aquest cas seria bastant complicat recuperar l'equilibri, així que és una velocitat suficient per l'aplicació que implementarem.

Veient les característiques per separat podem dir que és un mòdul bastant bo pel projecte, ja que a més no és un dispositiu massa car.

Veure datasheet's a 14.2.

- **Rodes**

Trobar unes rodes on és pogués adaptar l'eix dels motors va ser bastant complicat. Així doncs, em vaig decidir per comprar unes rodes inflables de 26 cm de diàmetre i fer un adaptador a mida per l'eix dels motors. Només tenia una idea en ment per les rodes, i és que fossin de més de 20cm de diàmetre.

En la figura 10.0.10 tenim una mostra de com són les rodes elegides.

Capítol 11

Implementació

11.1. Placa electrònica (PCB)

La placa principal del projecte funciona gràcies a un microcontrolador Atmega32u4, que juntament amb la IMU que conté el giroscopi IDG500 i l'acceleròmetre ADXL335, fan que el robot aconsegueixi mantenir l'equilibri, aquesta és la seva funció principal.

En la figura 11.1.1 veiem un diagrama esquemàtic de tot el sistema corresponent a la placa electrònica del robot.

11.1.1. Característiques elèctriques de funcionament

El primer aspecte a tenir en compte són els nivells de voltatge en què s'han d'alimentar els diferents dispositius que tenim a la placa, com l'acceleròmetre, el giroscopi, el pont H... Cada un d'ells està alimentat per diferents valors d'alimentació.

Per analitzar aquestes característiques tècniques dels diferents dispositius que utilitzem en el projecte tenim que el giroscopi i l'acceleròmetre són els únics que funcionen a 3V3, mentre que els altres dispositius, inclòs el microcontrolador treballen sense cap tipus de problema a un voltatge de 5V.

Per poder tenir 5V i 3V3 a partir d'una font de 12V utilitzem dos classes de reguladors. En el cas dels 5V utilitzem el LM7805, el qual entrega els 5V amb una corrent de 1A de màxim. En les especificacions del regulador observem que pot regular els 5V sense problema mentre el voltatge d'entrada sigui entre 7V i 20V, amb el que els nostres 12V seran perfectes per aquest regulador, ja que a més a més no tindrà problemes de temperatura. El giroscopi i l'acceleròmetre són els dos únics sensors del projecte que s'alimenten de 3V3, per aquesta raó el regulador que em vaig decidir a utilitzar és el 78M33, un regulador de voltatge en què els seus voltatge d'entrada poden estar entre 5,3V i 25V, amb una corrent de sortida de 500mA màxim.

11.1.2. Microcontrolador Atmega32u4

Per aquest projecte utilitzarem el microcontrolador Atmega32u4 ([ATM11, MTD11]), s'encarregarà de la conversió A/D de les senyals rebudes del acceleròmetre i del giroscopi, també de crear les senyals

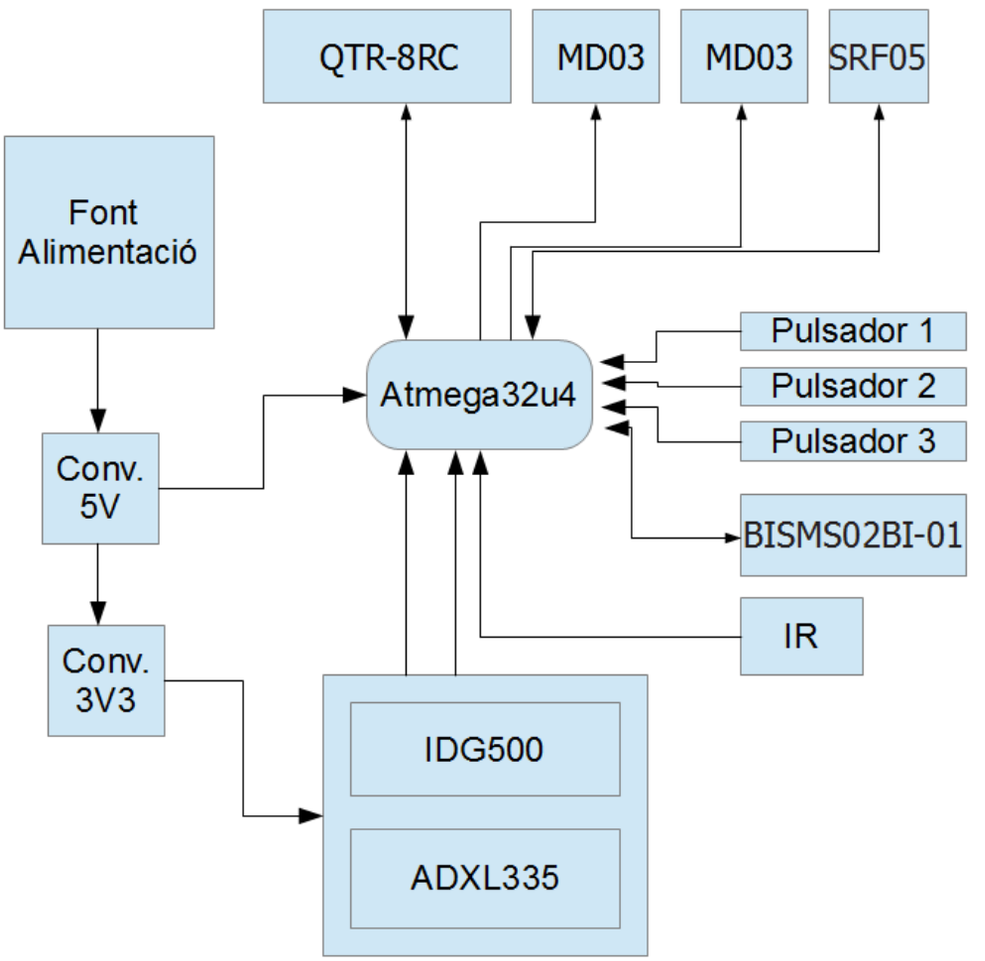


Figura 11.1.1: Diagrama esquemàtic PCB

pel funcionament del sensor d'ultrasons, la interpretació de les senyals rebudes pels diferents sensors i de la interrupció que executarà l'algoritme PID que traurà les senyals PWM pel moviment dels motors en funció de l'angle i la velocitat angular mesurats.

11.1.2.1. Descripció del microcontrolador

El Atmega32u4 és un microcontrolador (MCU) de 8 bits tipus RISC. Té un espai de memòria flash de 32Kbytés, però 2K bytés els utilitzem pel bootloader ja precarregat. A més a més, disposa de 1K Byté de memòria EEPROM i 2,5K Bytés de memòria SDRAM.

Disposa de 26 pins I/O, un oscil·lador extern de 16 Mhz i amb la possibilitat de tenir 4 timers. També té la possibilitat de tenir 6 interrupcions. A l'ésser un microcontrolador amb tecnologia CMOS té un baix consum d'energia, amb un rang d'operació de 2,7V a 5,5V d'alimentació i -40ºC a 85ºC de temperatura de treball.

Es capaç de treballar a una velocitat de fins a 16 MIPs, pot utilitzar un oscil·lador extern o intern que porta, que és de 8 Mhz. Sigui l'oscil·lador intern o extern l'utilitzat per treballar es pot utilitzar un prescaler de 1, 8, 64, 256 o 1024.

El microcontrolador Atmega32u4 té una sèrie de característiques pel que fa a perifèrics, disposa de 4 timers, un de 8-bits, dos de 16-bits i un últim de 10-bits de alta velocitat, un mòdul SPI, un mòdul UART amb Hardware Flow Control.

També té un conversor A/D de 10 bits amb capacitat per 12 canals d'entrada i un mòdul de control de motors amb 4 canals de sortida tipus PWM amb resolució programable de 2 a 16 bits i 6 canals de sortida tipus de PWM de alta velocitat amb una resolució programable de 2 a 11 bits.

11.1.2.2. Anàlisis dels recursos utilitzats pel microcontrolador

Pel projecte he utilitzat els següents recursos del microcontrolador (11.1.2).

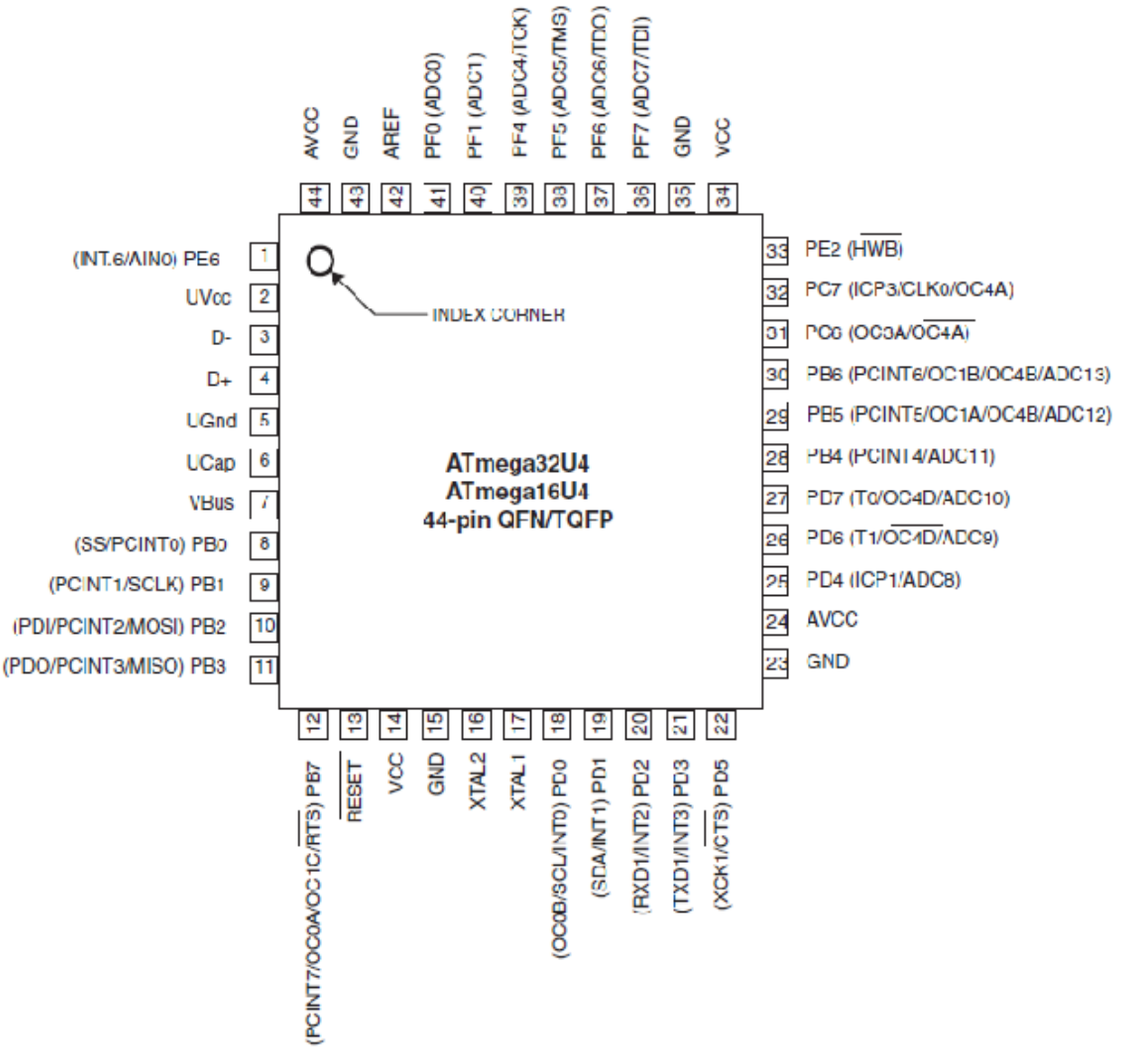


Figura 11.1.2: Pinout ATmega32U4

Veure datasheet en l'Annex, a la pàgina 119.

- **Conversor analògic-digital (Acceleròmetre i giroscopi)**

El conversor analògic-digital s'utilitza per les senyals rebudes de l'acceleròmetre i el giroscopi. D'on agafarem els valors de X i Z del acceleròmetre i el X del giroscopi.

Els pins utilitzats per aconseguir els valors d'aquestes senyals els mostrem en la taula 11.1.

- **Sensor ultrasons**

La informació enviada per crear la senyal dels ultrasons i seguidament rebre el rebot és enviada i rebuda des del pin que mostrem en la taula 11.2.

Nom	Pin Atmega32u4	Tipus	Funció
RF0 (X-ACC)	41	Entrada analògica	Entrada senyal de l'acceleròmetre eix X
RF4 (Z-ACC)	39	Entrada analògica	Entrada senyal de l'acceleròmetre eix Z
RF6 (X-RATE)	37	Entrada analògica	Entrada senyal del giroscopi eix X

Taula 11.1: Pins acceleròmetre i giroscopi

Nom	Pin Atmega32u4	Tipus	Funció
RE2(TRIGGER)	33	Entrada/sortida	Entrada i sortida sensor ultrasons

Taula 11.2: Pin Sensor ultrasons

- **Sensor IR**

Per poder detectar els documents que col·loquem en el robot tenim un sensor ir que ens enviarà una senyal. En aquest cas només ens envia un "1" si detecta un document i un "0" si no detecta res.

Veiem el pin en la taula 11.3.

Nom	Pin Atmega32u4	Tipus	Funció
RC6 (OUT)	31	Entrada digital	Detecció documents

Taula 11.3: Pin Sensor IR

- **Sensor seguir línies**

Amb el fi de poder detectar el màxim número de punts per la línia negra, utilitzarem 8 sensors. Els pins s'han de configurar com entrada/sortida per poder carregar els condensadors.

El procés que seguirem és:

1. Engegar els IR LED's
2. Establir les línies I/O com a sortida i amb un 1
3. Mantenir durant 10us perquè el condensador de 10nF es carregui.
4. Establir les línies I/O com a entrada i alta impedància
5. Mesurar el temps en què tarda el condensador a descarregar-se.

Veiem la taula de pins del sensor en la taula 11.4.

- **Bluetooth**

Amb el fi de poder enviar o rebre dades des de el mòdul Bluetooth, aquest, el connectarem en les entrades i sortides USART, per poder així configurar el mòdul com a dispositiu sèrie i poder utilitzar aquest protocol per enviar i rebre. Per saber més, mirar [23212, RS211].

Els pins que utilitzem pel mòdul Bluetooth són els de la taula 11.5.

- **Maneig dels motors**

Nom	Pin Atmega32u4	Tipus	Funció
RB0 (S1)	8	Entrada/Sortida	Connectat al Bit 1 de dades del sensor de la línia
RB1 (S2)	9	Entrada/Sortida	Connectat al Bit 2 de dades del sensor de la línia
RB2 (S3)	10	Entrada/Sortida	Connectat al Bit 3 de dades del sensor de la línia
RB3 (S4)	11	Entrada/Sortida	Connectat al Bit 4 de dades del sensor de la línia
RD0 (S5)	18	Entrada/Sortida	Connectat al Bit 5 de dades del sensor de la línia
RD1 (S6)	19	Entrada/Sortida	Connectat al Bit 6 de dades del sensor de la línia
RD4 (S7)	25	Entrada/Sortida	Connectat al Bit 7 de dades del sensor de la línia
RD5 (S8)	22	Entrada/Sortida	Connectat al Bit 8 de dades del sensor de la línia

Taula 11.4: Pins Sensor seguir línies

Nom	Pin Atmega32u4	Tipus	Funció
RD2 (TX)	20	RXD1	Rebre les dades enviades des de el mòdul Bluetooth
RD3 (RX)	21	TXD1	Enviar dades cap al mòdul Buetooth

Taula 11.5: Pins mòdul Bluetooth

El microcontrolador en base als valors de l'actitud calculats, genera unes senyals quadrades PWM variable pel control dels motors DC que es posicionaran d'acord l'angle mesurat. També controlarem la direcció dels motors a partir d'aquest valors calculats.

Aquestes senyals s'enviaran a través de 4 pins del microcontrolador tal com veiem en la taula 11.6.

Nom	Pin Atmega32u4	Tipus	Funció
RB6 (PWM1)	30	Generador d'ona quadrada	Control del moviment al motor 1
RB5 (PWM2)	29	Generador d'ona quadrada	Control del moviment al motor 2
RB4 (DIR1)	28	Sortida	Control de la direcció al motor 1
RB7 (DIR2)	12	Sortida	Control de la direcció al motor 2

Taula 11.6: Pins PWM

- **Conversor analògic-digital (Control Bateria)**

Utilitzarem un dels pins del microcontrolador per mesurar el voltatge de la bateria de 12V 7A, només mesurem aquesta ja que és la bateria que més consum té i el disseny de la placa el vaig fer amb la idea de tenir només una bateria.

Veiem el pin utilitzat en la taula 11.7.

- **Buzzer**

Per avisar d'un petit entrebanc en la trajectòria del robot utilitzarem un petit buzzer, connectat en un dels pins del microcontrolador. Quan detecta algun obstacle, utilitzem la interrupció del timer 4 per poder activar el buzzer.

Veiem el pin utilitzat en la taula 11.8.

- **Polsadors**

Nom	Pin Atmega32u4	Tipus	Funció
RF7 (BAT)	36	Entrada analògica	Mesurar nivell voltatge bateria

Taula 11.7: Pin Bateria

Nom	Pin Atmega32u4	Tipus	Funció
RB5 (BUZZ)	29	Sortida	Senyal sonora de proximitat

Taula 11.8: Pin buzzer

També he col·locat 3 polsadors per poder tenir un petit menú de configuració en el cas que no estigui disponible el mòdul Buetooth. Un dels tres polsadors serà l'encarregat de canviar la variable a modificar, i els altres dos s'utilitzaran per sumar o restar el valor de la variable.

Els pins que fan referència als polsadors els trobem a la taula 11.9.

Nom	Pin Atmega32u4	Tipus	Funció
RD6 (PD6)	26	Entrada	Polsador per menú alternatiu
RE6 (PE6)	1	Entrada	Polsador per menú alternatiu
RC7 (PC7)	32	Entrada	Polsador per menú alternatiu

Taula 11.9: Pin buzzer

11.2. Elèctric

Els sistemes de actuació elèctrics, ja siguin rotatius o lineals utilitzen l'energia elèctrica per transformar-la en energia mecànica.

El sistema elèctric del projecte és molt senzill, disposa de 2 bateries de 12V, una de 7A que alimenta tota la part de potència, és a dir, el pont H. Al ser l'alimentació dels motors i consumir més (pot arribar a consumir fins a ... A, de pic) he decidit que la bateria més potent seria ideal per alimentar aquesta part del projecte. En canvi, la segona bateria, en aquest cas de 1,2A, és l'encarregada d'alimentar la part electrònica del robot. Al tenir un consum molt menor els 1,2A són suficients per alimentar aquesta part.

Com veiem en l'apartat del disseny elèctric i tal com he comentat en l'apartat del Pont H anteriorment, la característica principal d'aquest apartat és la separació de les bateries un cop el robot està en marxa, així evitem que la part elèctrica i de potència s'uneixin i tinguem problemes en el Pont H.

Però un cop està el robot apagat, les bateries estan unides en sèrie, d'aquesta manera quan posem a carregar el robot les bateries es carreguen alhora. Per separar les bateries al engegar el robot utilitzem un switch doble, així podem tenir els dos estats.

Una altra característica és l'adaptació d'un fusible en el circuit de la bateria d'alimentació de la part de potència, he utilitzat un fusible de 10A, ja que si hi hagués un consum excessiu en aquella zona no circularia més enllà del fusible.

11.3. Protocol comunicació client/servidor via Buetooth

Per rebre i enviar diferents dades des de l'ordenador, em vaig decidir en crear un petit protocol de comunicació perquè el robot entengués les dades que li demanava canviar per poder realitzar diferents accions.

Els protocols estableixen una descripció formal dels formats que han de presentar els missatges per poder ser intercanviats entre diferents equips. Poden incloure senyalització, autenticació i detecció d'errors i la capacitat de correcció. Un protocol descriu la sintaxi, la semàntica i la sincronització de la comunicació i pot ser implementat en maquinari o programari, o ambdós. Els sistemes de comunicació utilitzen aquests formats predefinits (protocols) per intercanviar missatges. Un protocol descriu tant la sintaxi i la semàntica com la sincronització de la comunicació. Un llenguatge de programació descriu el mateix per als càlculs. Podem dir que hi ha una certa relació entre els protocols i llenguatges de programació: els protocols són per a la comunicacions el mateix que els llenguatges de programació són pels càlculs.

11.3.1. Especificació de protocol

La sintaxis està composta de 4 valors en hexadecimal.

C1 01 XX

On *C1 01* són els valors que indiquen al robot que voldrem canviar el valor d'alguna variable de configuració.

Els valors de *XX* varien depenen de la variable a modificar i si volen sumar o restar aquesta variable.

Per exemple:

Si enviem *C1 01 10*.

El robot rebrà via Buetooth aquesta comanda, llegirà *C1 01*, el que farà que ell sabrà que volem canviar el valor d'una variable. Seguidament llegirà el valor de *10*, on el primer valor '1' tenim que canviarà el valor de la variable KP. I els pròxims 4 bits, en aquests cas '0' en hexadecimal significarà que voldrem restar el valor de KP. A part de la sintaxis anterior, també tenim una altra sintaxis, en aquest cas serà la sintaxis que utilitzem per controlar el robot.

La sintaxis de control del robot és la següent:

5A 41 XX

La raó d'utilitzar una sintaxis diferent només és la de separar el control i la configuració. En aquest cas funciona igual que la anterior. Quan el robot rep *54 41* sap que volem començar a moure el robot i els 8 bits restants poden tenir només 5 valors, els que faran que el robot es mogui endavant, endarrere, esquerra, dreta o parat.

Per exemple:

Si enviem *5A 41 41*

El robot rebrà via Buetooth aquesta comanda, llegirà *5A 41*, el que farà que ell sabrà que volem moure el robot. Seguidament llegirà el valor de *41*, on llegirà directament els 8 bits i farà que el robot, canviï el valor de la variable despla per moure's endavant.

Part del codi utilitzat:

```
if (uart_available() >= 1 && connectat == 1)
        {
                first = uart_getchar();

                if(first == 0xC1)
                {

                        opcio = uart_getchar();
                        varia = uart_getchar();

                        //Opció Ajustar

                        if ( opcio == 0x01 )
                        {

                                //Variem KP

                                if ( (varia&0x0F) == 1 )
                                {

                                        if ( (varia&0xF0) == 0x10 )
                                                KP+=0.1;
                                        else
                                                KP-=0.1;

                                }
                                .
                                .
                                .
        }
```

11.4. Control del sistema

En aquest capítol veurem la programació del Atmega32u4, en base a la configuració bàsica dels diferents mòduls utilitzats en el programa. A més s'indicarà com es va realitzar la implementació de l'algoritme PID per fusionar les senyals del giroscopi i de l'acceleròmetre i així poder aconseguir que la mesura dels valors trobats sigui el més acertada possible.

11.4.1. Programació de l'Atmega32u4

Ara veurem els principals criteris aplicats a la configuració dels diferents mòduls utilitzats per la creació del microcontrolador

Treballarem amb un cristall extern al microcontrolador de 16Mhz (ja integrat en la placa MT-DB-U4). Amb aquesta freqüència de treball tenim que el període del cicle del robot serà de:

$$T = \frac{1}{Fosc} = \frac{1}{16000000} = 0{,}0000000625s = 0{,}0625us \text{ (clock de rellotge)}$$

Aquest valor serà molt important, sobretot en els càlculs dels períodes de temps.

11.4.1.1. Mòdul de sortida PWM

El timer 1 s'utilitzarà per crear la senyal pwm, aquest té 3 sortides independents pel PWM, que són el OC1A, OC1B i OC1C. Aquest és un timer de 16 bits.

De les diferents opcions per la generació del PWM s'utilitzarà el mode de funcionament Phase and Frequency Correct PWM Mode, ja que permet generar senyals de PWM de alta resolució amb fase i freqüència correcta. El comptador conta repetidament des de BOTTOn (0x0000) fins a TOP i després des de TOP fins a BOTTOM. En el mode de operació no invertit del Compare Output, la sortida OC1x es posa a 0 quan hi ha igualtat entre TCNT1 i OCR1x mentre comptem de manera ascendent i a 1 quan comptem de manera descendent. En el mode d'operació invertida els valors de sortida s'inverteixen. Degut a la simetria de la senyal PWM generada en els modes d'operació que utilitzen la tècnica de la doble pendent, aquests modes són els més utilitzats per les aplicacions de control de motors.

S'utilitzarà el mode d'operació 8, pel que s'ha de configurar els bits WGM13:0, que trobarem als registres TCCR1A i TCCR1B, de la següent manera:

WGM13 = 1
WGM12 = 0
WGM11 = 0
WGM10 = 0

En aquest mode de operació el valor TOP es guarda en el registre ICR1. S'utilitzarà la sortida OC1A i OC1B no invertida, pel que s'han de configurar els bits COM1A1:0 que trobem en el registre TCCR1A, de la següent manera:

COM1A1 = 1
COM1A0 = 0

Es important tenir el bit corresponent en el registre DDR al PB6 i PB5 han d'estar configurats com a sortida.

A continuació mostrem el càlcul de la freqüència en què crearem la senyal PWM, vaig buscar que la freqüència tingués unes característiques concretés, que són:

1. Una senyal no audible per les persones humanes.
2. Els mosfet treballin dintre dels paràmetres adients.
3. Una bona resolució

$$Fpwm = \frac{Fosc}{2} \cdot Prescaler \cdot TOP = \frac{16000000}{2} \cdot 1 \cdot 511 = 15655{,}6Hz = 15{,}6Khz$$

On TOP és el valor que s'assigna al registre ICR1 i N representa el valor del preescaler, en el nostre cas utilitzem un precaler a 1.

Provant diferents configuració vaig arribar a la conclusió que 15,6Khz era la freqüència adient, ja que no es audible i els mosfets treballen bé amb ella, a més això em dóna una resolució de 9bits, cosa que està molt bé per la seva funció.

Seguidament veurem la funció que inicialitza el PWM i la funció on ja donem els valor als motors.

```
// Inicializa PWM
void Pwm_Init( void )
{

        // Puertos PWM como salida
        sbi(DDRB,5); // pwmL
        sbi(DDRB,6); // pwmR

        timer1PWMInitICR(511); //511 9bits

        //Engeguem PWM
        timer1PWMAOn();
        timer1PWMBOn();

        timer1PWMASet(0); // 0-511
        timer1PWMBSet(0); // 0-511

        sbi(DDRB,4); // Dirección MotA
        sbi(DDRB,7); // Dirección MotB
        sbi(PORTB,4);
        sbi(PORTB,7);

}
```

En la funció Pwm_Init inicialitzem el timer 1 per a que funcioni com a PWM.

```
void run_motors(u16 motL, u16 motR)
{


        // Establim direcció dels motors
        if (motR>=512)
        {
                motR = motR-512;
                sbi(PORTB,4); //endavant

        } else {

                motR = 512 - motR;
                cbi(PORTB,4); // endarrera
        }


        if (motL>=512)
        {
                motL = motL-512;
                sbi(PORTB,7); //endavant

        } else {
```

```
                motL = 512 - motL;
                cbi(PORTB,7); // endarrera
        }

        // Valor càlculat als motors
        timer1PWMASet(motL);
        timer1PWMBSet(motR);
}
```

En la funció run_motors dóna els valors calculats als motors, dependent del valor sabem la direcció en què anem i en quina velocitat. Els valors de motL i motR venen limitats, ja que no volem que ens passi de -0.6 i 0.6. La raó de la limitació és que podem tenir en algun moment algun càlcul erroni.

11.4.1.2. Conversor Analògic/Digital

- **Acceleròmetre i giroscopi**

El Conversor Analògic/Digital del Atmega32u4 és per aproximacions successives amb una resolució de 10 bits (és a dir, entre 0 a 1023). El ADC es connecta a un multiplexor de 8 canals analògics. El ADC està compost per un circuit de mostreig i retenció que assegura que el voltatge d'entrada al ADC es mantingui constant durant al conversió.

El ADC té un pin per la font de voltatge separat, AVCC. L'AVCC no pot passar de +-0,3V dels Vcc. També té un voltatge intern de referència nominal de 2,56V(és a dir, entre 0 i 2,56V) de AVCC. En el nostre cas utilitzarem el voltatge de referència intern, ja que el voltatge de sortida del giroscopi i l'acceleròmetre màxim es de 2,5V. Per utilitzar aquest voltatge em de canviar del registre ADMUX els bits REFS1 i REFS0:

REFS0 = 1
REFS1 = 1

Per defecte, la circuiteria per aproximacions successives necessiten d'una freqüència de rellotge d'entrada entre 50kHz i 200kHz per tenir la màxima resolució. Si volem una resolució més baixa de 10 bits, la freqüència del rellotge d'entrada del ADC pot ser més gran de 200kHz, però en el nostre cas volem el màxim de resolució, així que busquem una freqüència de rellotge entre els valors d'entrada anomenats anteriorment.

Tenim un cristall de 16Mhz i el ADC té uns factors de divisió de 2, 4, 8, 16, 32, 64 i 128 hem decidit que la freqüència correcta la trobarem amb el factor de divisió 128, ja que ens dóna 125kHz, dintre dels valors que volem.

$$Freqüència = \frac{16000000}{128} = 125000Hz = 125Khz$$

Per fer que el ADC del nostre microcontrolador funcioni amb aquesta freqüència em de canviar els bits ADPS2, ADPS1 i ADPS0 del registre ADCSRA.

ADPS2 = 1
ADPS1 = 1
ADPS0 = 1

El prescaler inicia el seu conté des de el moment en què el ADC s'engega col·locant el bit a un a ADEN del registre ADCSRA. El prescaler el manté corrent mentre ADEN està a 1, i es reinicia quan ADEN està a nivell baix.

El canvi d'aquests bits ens determinarà el factor de divisió entre la freqüència del XTAL i l'entrada del rellotge al ADC. El que ens determina aquesta freqüència és el temps que tardarà en fer la lectura del conversor.

Com em vist en l'apartat anterior, els mòduls més importants i que utilitzen el ADC són el giroscopi i l'acceleròmetre. Per seleccionar quin canal d'entrada volem connectar al ADC tindrem que canviar els bits MUX4, MUX3, MUX2, MUX1 i MUX0 del registre ADMUX, per exemple:

Si volem llegir X-ACC, que és l'entrada de la senyal del acceleròmetre eix X i que tenim connectat al pin 41 del Atmega32u4 i que fa referència al ADC0 posarem els bits:

MUX4 = 0
MUX3 = 0
MUX2 = 0
MUX1 = 0
MUX0 = 0

Quan acaba la conversió (ADIF, que es el bit que trobem en el registre ADCSRA i que funciona com a bandera d'interrupció del ADC, es alta), el resultat de la conversió es pot trobar en el registre de resultat ADC. Per una sola conversió, el resultat és:

$$ADC = ADC - offset$$

$$Vin = ADC \cdot \left(\frac{Vref}{1024}\right)$$

El que fem és restar el resultat de l'offset, el offset és el valor 0, ja que podem tenir valors negatius i positius. És a dir, tenim que amb el voltatge de referència tenim valor entre 0 i 2,56V amb un mostreig de 10bits, cosa que fa que entre aquests dos valors hi hagin 1024 divisions, on cada divisió és de 0,0025V. Per exemple, en el giroscopi tenim que a 0^{o}/s el ADC ens dóna un valor de giroscopi de 520, aquest serà el nostre offset.

Com que al datasheet del giroscopi ens diu que màxim es mourà a +-500^{o}/s amb una sensibilitat de 0,002V/^{o}/s i sabem que el nostre ADC té una divisió per passos de 0,0025V i calculem que el nostre offset 520 equival a 1,3V. Per tant, si fem:

$$V = 500^{o}/\text{s} \cdot 0{,}002V/^{o}/s = 1V$$

Això ens diu que tindrem una variació de 1V positiu i 1V negatiu. Cosa que fa que tinguem una variació de voltatge de 0,3V – 1,3V – 2,3V. Veiem que el valor màxim que ens pot donar el giroscopi és de 920 i el mínim 120. Pel que observem que tenim una variació de 400 valors positius i 400 valors negatius.

Un cop fets aquests càlculs anteriors volem arribar a saber en quants ^{o}/s ens movem. Si sabem que la màxima variació és de 400 i que quan arribem a aquest valor màxim tindrem 500^{o}/s calcularem quants ^{o}/s tenim per variació:

$$Variació = \frac{500}{400} \cdot 1 = 1{,}25^{o}/s \text{ per variació}$$

Un cop tenim les variacions, volem identificar quan la velocitat angular és positiva i quan negativa. Sabem que 520 és el nostre offset, és a dir, els nostres $0^{o}/s$ i que 400 seran els passos màxims positius i negatius. Per tant, del valor que ens retorni el ADC quan fem una lectura del giroscopi l'hi restarem 520. Per exemple, si ens dóna 920:

$$Gyro_X = 920 - 520 = 400$$

Per tant tenim que anem en direcció positiva. En canvi, si el valor retornat és 120:

$$Gyro_X = 120 - 520 = -400$$

En aquest cas anem en direcció negativa.

Un cop tenim tots els càlculs tenim com a resultat:

$$gyro_x = (ADC - offset) \cdot 1{,}25$$

Així ja tenim els $^{o}/s$ en què ens mourem

Tenim el mateix amb l'acceleròmetre, però en aquest cas no busquem $^{o}/s$, sinó que medeix acceleració, i una forma de de calcular acceleració es g's, amb aquest resultat:

$$\text{acc\_x\_g} = \left((float)\,(ADC - offset) \cdot \frac{(\frac{2{,}56}{1024{,}0})}{0{,}3} \right);$$

On tenim que la variació de voltatge és entre 0,6 i 2,40V, i 1,5V tenim que són els nostres 0 g's. Amb una sensibilitat de 0,3V/g.

- **Bateria**

A diferència dels sensors anteriors, per saber el voltatge que tenim a la bateria no utilitzem el voltatge de referència intern, ja que així tenim més resolució, per tant s'ha utilitzat un Vref extern.

Per això, quan volem accedir al ADC de la bateria el primer que em de fer és canviar els bits REFS1 i REFS0 del registre ADMUX, quedant de la següent manera:

REFS1 = 0
REFS0 = 1

Activant així el Vref extern.

Per poder saber en tot moment el voltatge de la bateria (només de la bateria de 12V 7A, ja que és la que més consum té utilitzem el divisor de tensió de la figura 11.4.1.

On R2 té el valor de 10K i R3 de 4K7, i l'entrada "12V" és Ventrada. Veiem que l'esquema del divisor de tensió és molt simple. Es tracte de dos resistències en sèrie connectades de tal manera que l'entrada Ventrada es connecta als extrems d'aquestes i obtenim la tensió de sortida BAT d'un extrem i del punt mig entre les dos resistències. La resistència que comparteixen els dos és R3 i l'altra és la R2.

La fórmula per calcular BAT en funció de l'entrada "12V" i les dos resistències és:

$$BAT = \left(\frac{R3}{R3} + R2\right) \cdot Ventrada$$

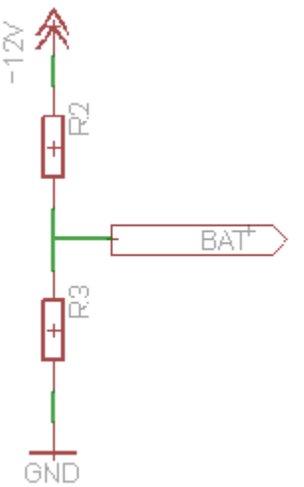


Figura 11.4.1: Divisor de tensió

A BAT tindrem un valor entre 0 i 5V, i el ADC ens convertirà aquests valor a digital. Per exemple:

0V = 0
5V = 1023

Un cop tenim el resultat del ADC utilitzarem la següent fórmula per saber el nivell de voltatge de la bateria.

$$Nivell_V = ADC \cdot \frac{\left(\frac{5}{1024}\right)}{0{,}32}$$

Un cop convertit el valor d'entrada volem saber a quin voltatge entre 0 i 15,5V equival, per tant, sabem que a cada 0,32V que tindrem a l'entrada del ADC serà 1V a la bateria.

11.4.1.3. Timer 0

En aquest apartat veurem com configurar i utilitzar els registres del Timer 0 per utilitzar-lo com a interrupció del cicle principal.

Els registres del Timer 0 (TCNT0) són de 8 bits. El modo de funcionament que em triat és el mode normal, que és el mode més simple.

Per començar, hem decidit que volem que la nostra interrupció tingui una durada de 10ms, és a dir, 0,01s. Com hem calculat anteriorment, el nostre cristall tarda 0,0625us en fer un cicle. Observem que és una velocitat massa alta per nosaltres, així que utilitzarem un prescaler per disminuir aquesta velocitat de cicle.

El prescaler del timer 0 es pot configurar a 8, 64, 256 i 1024. Com que la velocitat que volem anar és realment baixa provarem amb el prescaler a 1024. Per tant:

$$TempsAmbPrescaler = 0{,}0000000625s \cdot 1024 = 0{,}000064s$$

Observem que la velocitat més baixa a que podem anar és lleugerament major a la que necessitem, però com no podem anar més lents utilitzarem aquest prescaler. Per configurar-lo em de canviar els bits CS02, CS01 i CS00 del registre TCCR0B, inicialitzant-los de la següent manera:

CS02 = 1
CS01 = 0
CS00 = 1

Tenim que el timer 0 és de 8bits, això ens diu que el màxim de tics del timer són 256, per tant des de que comença a contar el timer 0 fins que arriba a 256 tarda 0,000064s.

Per arribar als nostres 0,01s utilitzant el prescaler a 1024 necessitem inicialitzar el prescaler en un valor superior a 0. Així doncs:

$$Tics = \frac{0{,}01s}{0{,}000064s} = 156{,}25tics$$

El número de tics necessaris són 156,25 tics, que són els que necessita perquè la duració del timer sigui de 10ms.

Per tant, volem que el timer salti a 156,25 i no a 256. Així, inicialitzarem el nostre timer a 100. Per inicialitzar el timer a 100 tenim que establir el registre TCNT0 a 100.

Un cop tenim el timer funcionant només ens falta activar la interrupció, on només tenim que modificar el bit TOIE0 del registre TIMSK0.

TOIE0 = 1

Configurat el timer 0 per a que desbordi cada 10ms, tenim que activar la interrupció de sistema per a que activi la funció del cicle principal. Aquesta interrupció del sistema és:

```
SR(TIMER0_OVF_vect)
        {
                        .
                        .
                        (cicle principal)
                        .
                        .
        }
```

11.4.1.4. Ultrasons

Per mesurar les distàncies dels objectes que ens poden obstruir el pas utilitzem el mesurador de distància SRF05, capaç de mesurar distàncies de fins a 4 metres.

Disposa de 2 modes de operació:

- Mode 1:

Aquest mode utilitza patilles separades, una per aplicar el puls d'inici o Trigger i l'altra per llegir l'amplada del puls del ECO mesurat. Per seleccionar aquest mode només hem de deixar el pin "Mode" sense connectar. Veure figura 11.4.2.

Tal i com mostrem en el diagrama de temps 11.4.3, aquest mode és molt senzill. Externament aplicarem un puls de 10us de duració mínima per iniciar la sentència. En aquest moment el mòdul transmet

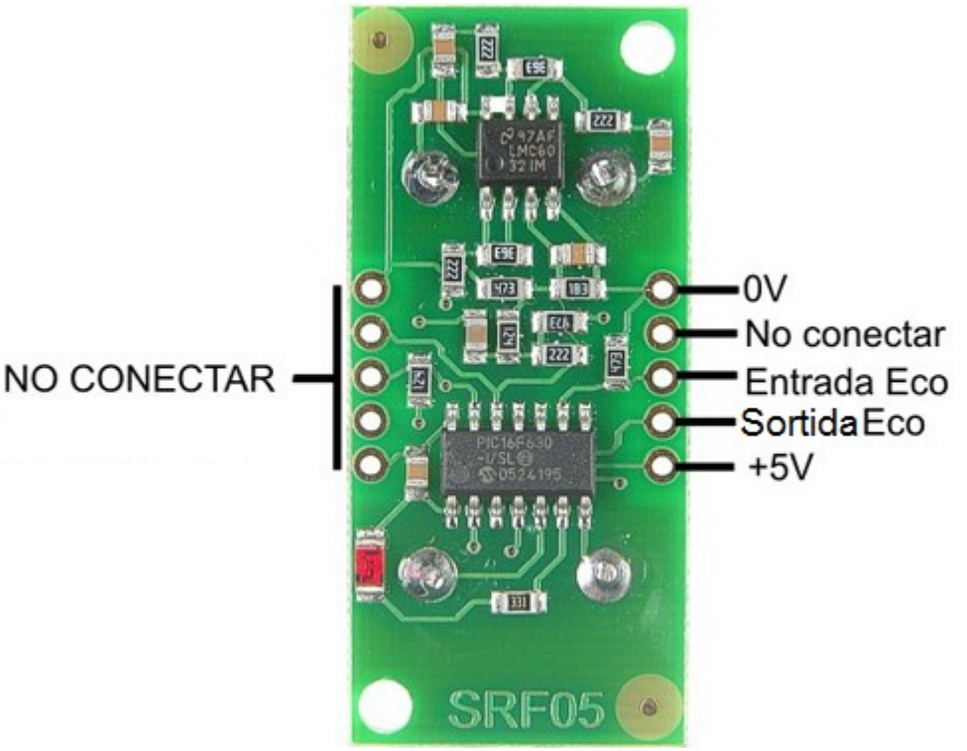


Figura 11.4.2: Mode 1 Ultrasons

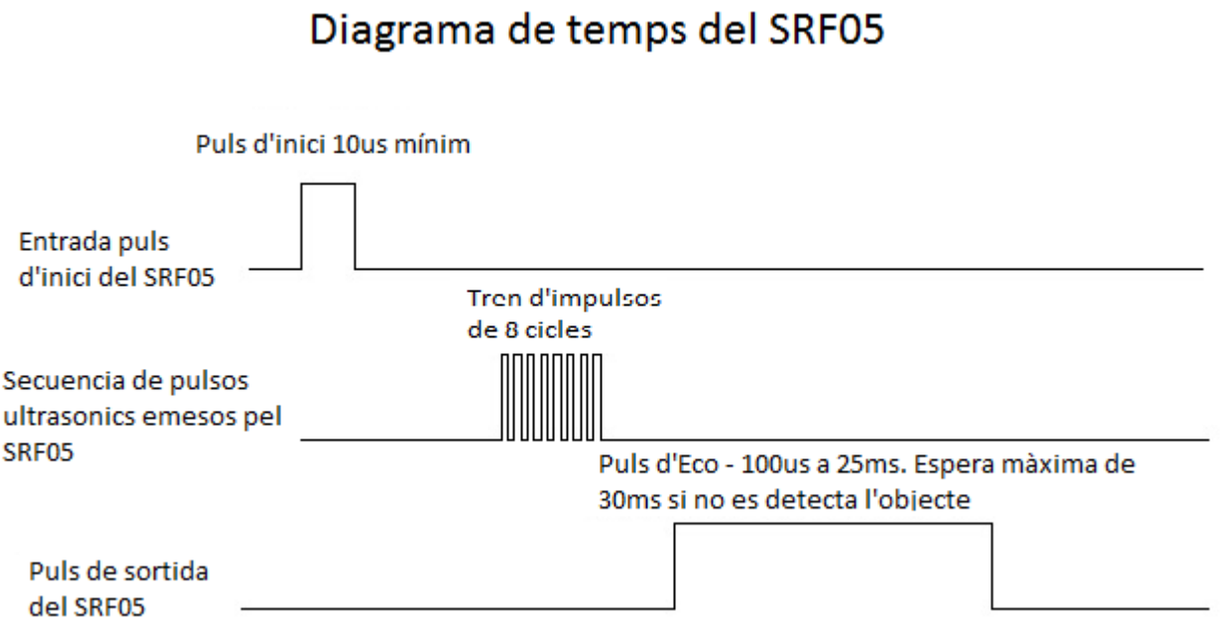


Figura 11.4.3: Diagrama de temps mode 1 SRF05

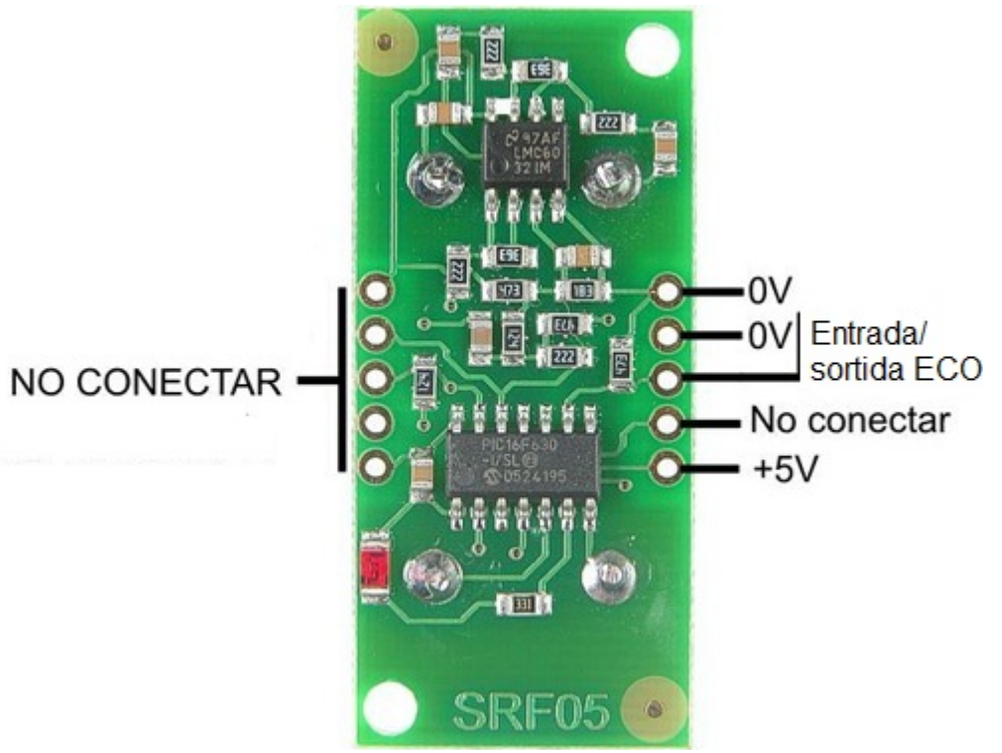


Figura 11.4.4: Mode 2 Ultrasons

un tren de pulsos de 8 cicles a 40Khz. Tot seguit la senyal de sortida passa a nivell "1". Quan la càpsula receptora rep la senyal transmès com a conseqüència d'haver rebotat (eco), aquesta sortida passa de nou a nivell "0". Nosaltres mesurarem la duració del puls d'aquesta senyal, és a dir, el temps que la senyal del eco és manté a "1".

Un cop fet això, hem de deixar al mòdul un temps perquè s'estabilitzi, aquest temps ha de ser un mínim de 20ms entre el moment que la senyal d'eco passa a "0" i un nou puls que inicii una nova mesura.

Això permet realitzar mesures cada 50ms. La duració del puls d'eco de sortida varia entre 100μs i 25ms, en funció de la distància entre el mòdul i l'objecte. La velocitat del so és de 29.15μs/cm que, com realitza un recorregut d'anada i tornada, és 58.30μs/cm. Així doncs el rang mínim que podem mesurar són 1.7cm (100μS/58) i el màxim 431cm (25mS/58).

- Mode 2:

A diferència del mode 1, aquest mode permet utilitzar només un pin per generar la senyal de trigger i també per realitzar la mesura de l'ample de puls de la sortida de l'eco, el que ens estalviem pins del microcontrolador.

Per utilitzar aquest mode només tenim que connectar el pin "Mode" amb GND (Veure figura 11.4.4). La senyal de l'eco apareixerà llavors al mateix pin que hem aplicat la senyal de trigger. Aquesta patilla la tenim que configurar primer com a sortida per generar la senyal de trigger i després com entrada per llegir la duració de l'eco. Veiem el diagrama de temps en la figura 11.4.5.

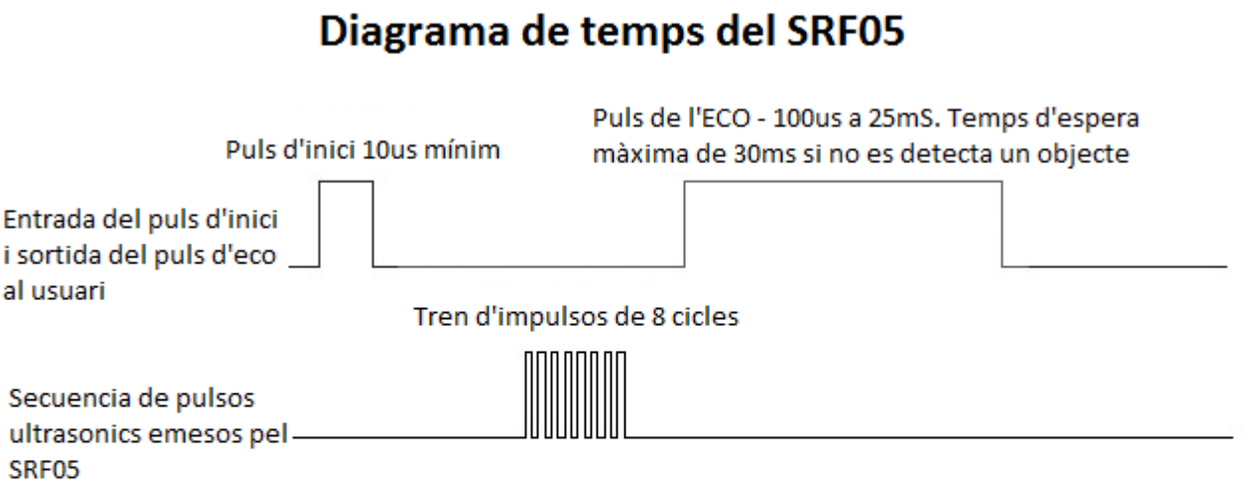


Figura 11.4.5: Diagrama de temps mode 2 SRF05

El primer mode és el més fàcil d'utilitzar des de el punt de vista del programador, ja que comptem amb terminals independents per l'entrada i la sortida, tot i això, des de el punt de vista de connexions el mode 2 és el millor, ja que només utilitzarem un pin del microcontrolador. Per això, pel projecte el mode 2 és la millor opció, ja que només tindrem un cable per controlar-lo.

Tot seguit veurem la funció creada per llegir el mòdul SRF05. Primer configurem el pin com a sortida i donem un puls de 10us. Un cop em donat aquest puls canviem el pin configurant-lo com entrada i a "0" i inicialitzem el timer 3. Esperem que la línia és posi a 1, mentre esperem deixem el timer 3 en marxa amb un límit, d'aquesta manera ens assegurem que esperem el temps necessari per a que passi a "1". Un cop tornem a tenir un "0" voldrà dir que ha trobat un objecte o que no hem trobat res. Ara ja tenim el temps que ha passat amb microsegons. Per calcular els centímetres utilitzem la següent fórmula.

$$Temps = \left(\frac{ticksTimer3 \cdot 0{,}5}{58}\right)$$

On multipliquem el temps per 0.5 i al dividir entre 58 tenim ens centímetres, ja que 1cm són 58us.

```
u16 Read_ultrasons (void)
{

        u16 ticks_timer3;


                timer3SetPrescaler( TIMER_CLK_DIV8 );
                sbi(DDRE,2); //Sortida
                sbi(PORTÉ,2); //posem a 1

//Donem puls de 10 us perquè comenci a enviar
                _delay_us(10);

                cbi(DDRE,2);
                cbi(PORTE,2);
//Esperem que la linia es posi a 1
                TCNT3 = 0;
                while(!(inb(PINE&4))){

                        if(TCNT3 == 60000) return 0;

                }
//Quan es 1 iniciem el timer
                //TCNT3H = 0;
                //TCNT3L = 0;
                while(inb(PINE&4));
//Esperem fins que torni a 0 i llegim el timer
                ticks_timer3 = TCNT3;
                sbi(TIFR3,0);
                return (u16)((ticks_timer3 * 0.5) / 58.0);

}
```

Un cop tenim els centímetres, he decidit que pel projecte el robot es pararà i sonarà el buzzer quan detecti un objecte a 65 cm.

11.4.1.5. IR

Per detectar si tenim algun document utilitzem el sensor GP2Y0D810Z0F. És un sensor de proximitat que s'encarrega de dir-nos si tenim algun document a la nostra safata. Està preparat per mesurar fins a una distància de 10cm. Aquesta distància va fer que em detectés l'altra banda de la bandeja sense tenir cap document, així doncs per solucionar aquest problema només vaig tenir que posar un adhesiu negre a l'altra banda perquè no detectés el sensor el retorn.

Per utilitzar-lo només tenim que connectar l'alimentació de 5V, ja que pot ser alimentat entre 2.7V i 6.2V sense problema, la massa i la sortida al dispositiu de control.

Aquest sensor ens mostra a la seva sortida una senyal digital a partir d'una senyal enviada pel díode i rebuda per un fotodetector. Com és basa amb el principi de triangulació, ni la temperatura ambient ni altres factors poden alterar la mesura de la distància del document.

La funció creada per fer la lectura d'aquest sensor és molt més senzilla que la creada pel sensor d'ultrasons, ja que només em de llegir quin valor en digital ens retorna el pin on està connectat. Així doncs:

```
u08 ReadIR (void)
{
        return inb(PINC&64);
}
```

Per saber el valor que ens retorna només hem de cridar la funció anterior de la següent manera:

```
if (ReadIR()>>6 == 1)

                papers = 1;
        else

                papers = 0;
```

Veiem que si en el bit corresponent tenim un 1 significarà que tenim un document, en el cas contrari no tenim document.

11.4.1.6. Seguidor de línies

El sensor QTR-8RC està dissenyat per ser utilitzat com a seguidor de línies, justament la finalitat que tenim en el projecte. Aquest mòdul està compost de 8 emissors i fotodetectors. Cada fotodetector utilitza un circuit de descarrega de condensador que permet a l'entrada del microcontrolador agafar la mostra analògica creada a partir de la reflexió mesurada pel temps de descarrega del condensador. Un menor temps de descarrega del condensador vol dir que la reflexió és més gran, per tant, en el nostre cas, tenim un color clar, com podria ser un blanc.

Totes les sortides són independents, però els díodes estan col·locats en parelles per poder reduir a la meitat el consum del QTR-8RC

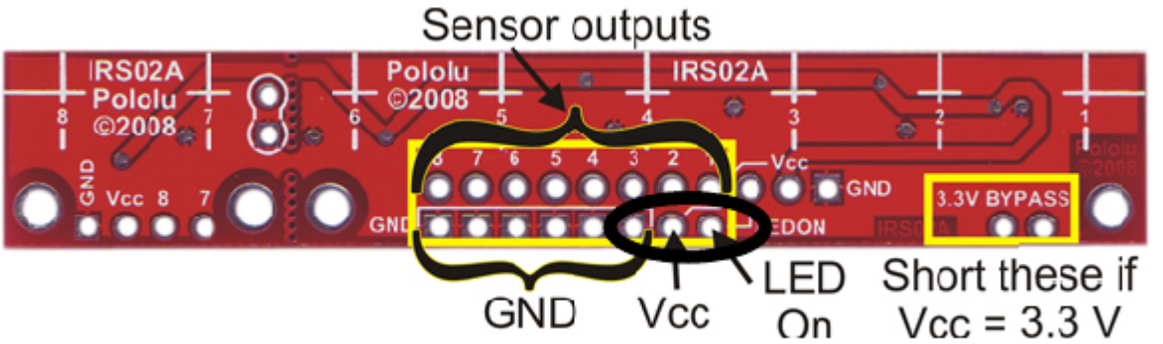


Figura 11.4.6: Sensor QTR-8RC

El primer que em de decidir a l'hora de configurar aquest sensor és si volem que els leds estiguin sempre actius o que els controlem a partir del microcontrolador. Pel projecte crec que la millor decisió és la de mantenir sempre actius els leds, d'aquesta manera alliberem al microcontrolador a l'hora de realitzar una nova tasca i sempre controlarem en quin terreny ens trobem. Per mantenir sempre actius els díodes leds sol em de fer un pont entre el pin LEDON i Vcc tal com veiem en la figura 11.4.6.

Un cop fet això, per activar aquest mòdul hem de seguir els següents passos:

1. Engegar els IR LED's.
2. Establir les línies I/O com a sortida i amb un 1.
3. Mantenir durant 10us perquè el condensador de 10nF es carregui.
4. Establir les línies I/O com a entrada i alta impedància.
5. Mesurar el temps que tarda el condensador a descarregar-se.

La funció corresponent creada per fer aquesta tasca és la següent:

```
void ReadIR_line (void)
{
        u08 sens_linia;

        if(prescaler_linia) timer3SetPrescaler( TIMER_CLK_DIV256 );
        else timer3SetPrescaler( TIMER_CLK_DIV1024 );

                DDRB |= 0x0F; //Sortida
                PORTB |= 0x0F; //Posem a 1

                outb(DDRD,0x33);
                outb(PORTD,0x33);
                outb(TCNT3, 65279);
                while(inb(TCNT3) < 3);

                DDRB &= 0xF0; //Entrada
```

```
                sbi(PORTB,0);
                sbi(PORTB,1);
                sbi(PORTB,2);
                sbi(PORTB,3); //posem a 1, pullup

                outb(DDRD,0x00);
                outb(PORTD,0x33);
                sbi(TIFR3,0); //Entrem 1 logic per reiniciar el flag
                outb(TCNT3, 65279);

                sens_linia = 0x00;


                while(!(inb(TIFR3&1)))
                {
                        sens_linia =  (inb(PIND&32)<<2)
                        | (inb(PIND&16)<<2)
                        | (inb(PIND&2)<<4) | (inb(PIND&1)<<4)
                        | inb(PINB&8) | inb(PINB&4)
                        | inb(PINB&2) | inb(PINB&1);
                }
                return sens_linia;

}
```

Com observem en la funció, el primer que fem és configurar els pins com a sortida i a "1". Després utilitzem el timer 3 per mantenir durant almenys 10us aquest "1" i per tant carregar els condensadors. En el nostre cas esperem 125us, ja que amb 10us els condensadors no es carregaven del tot. Un cop carregats els condensadors configurem els pins com a entrada i pullup. Reiniciem la bandera del timer 3 i comencem a contar el temps de descarrega. Un cop TIFR3 desborda crearem una variable u08 amb els valors de cada pin, és a dir, si durant la descarrega un dels fotodetectors detecta un negre aquest passarà a ser "1" abans que el TIFR3 desbordi. Així doncs, en el cas que els 4 fotodetectors centrals detectin un negre i els altres un color clar la variable sens_linia serà "00111100"

Un cop tenim aquest valor, utilitzarem aquest algorisme per mantenir-nos en la línia negra:

```
if (( sens_linia < 7 && sens_linia > 0 && despla != 0){

                girar = 0.03;
}

else if ( sens_linia == 6 ){

                girar = 0.04;

}

else if ( sens_linia == 48 ){

                girar = -0.03;
```

```
}

else if ( sens_linia == 96 ){

                girar = -0.04;

}

else if ( sens_linia == 255 ){

                girar = 0.0;

}
```

Aquest algorisme és molt senzill, el qual segons el valor de sens_linia girem cap a la dreta o l'esquerra, i ens mantenim endavant en el cas que sens_linia sigui 255, és a dir, "11111111"

Tenim també una funció per calibrar la resposta del sensor, on el que farem serà canviar el valor del prescaler del timer 3 depenent del color del terreny. Això ho fem perquè si el terreny era marronós el sensor creia que era un negre, ja que el timer desbordava abans que amb color clar. Lo adient és que el terreny sigui clar i una línia negra, per poder diferenciar millor. Tot i això, s'ha creat aquesta funció per si el terreny no pot ser un color clar.

La funció per calibrar el sensor depenent del terreny és el següent:

```
void Calibrar_sigue (void)
{
        u08 sens_linia;
        //Autocalibrat

                timer3SetPrescaler( TIMER_CLK_DIV1024 );
                DDRB |= 0x0F; //Sortida
                PORTB |= 0x0F; //Posem a 1

                outb(DDRD,0x33);
                outb(PORTD,0x33);
                outb(TCNT3, 65279);
                while(inb(TCNT3) < 3);

                DDRB &= 0xF0; //Entrada
                sbi(PORTB,0);
                sbi(PORTB,1);
                sbi(PORTB,2);
                sbi(PORTB,3); //posem a 1, pullup

                outb(DDRD,0x00);
                outb(PORTD,0x33);
                sbi(TIFR3,0); //Entrem 1 logic per reiniciar el flag
                outb(TCNT3, 65279);
                sens_linia = 0x00;
```

```
                while(!(inb(TIFR3&1)))
                {
                        sens_linia =  (inb(PIND&32)<<2)
                        | (inb(PIND&16)<<2)
                        | (inb(PIND&2)<<4) | (inb(PIND&1)<<4)
                        | inb(PINB&8) | inb(PINB&4)
                        | inb(PINB&2) | inb(PINB&1);
                }
                if(sens_linia != 0) prescaler_linia = 1;
                else prescaler_linia = 0;
}
```

Podem veure que els prescaler utilitzats són 256 o 1024. En cas que la superfície sigui clara agafarà un prescaler a 256, ja que la descarrega serà més ràpida, en cas contrari agafarà 1024.

11.4.1.7. Bluetooth

Per la comunicació entre els dos sistemes, robot i sistema de control de l'usuari he elegit un sistema inal·làmbric estàndard, aconseguint així reduir costos de desenvolupament i més compatibilitat entre dispositius al ser estàndard. S'ha elegit aquesta capacitat inal·làmbrica per mantenir la estètica inicial del robot sense cables que surtin d'ell (només el d'alimentació), a més proporciona una bona i nova funcionalitat al sistema com el control remot, el monitoratge o el de configuració a distància.

Hem elegit així un sistema de comunicació Buetooth per les següents característiques:

1. És un sistema de comunicació estàndard que ens dóna la possibilitat de comunicar-nos des de qualsevol dispositiu portable que tingui aquesta tecnologia amb el robot.
2. Es un sistema de comunicació molt adequat per aquest tipus d'aplicació i també el fa més econòmic en costos del component.
3. Existeixen 3 classes segons el seu radi de cobertura, sempre a terreny pla.

- Classe 1: Pot transmetre a distàncies de fins a 100m
- Classe 2: Pot transmetre a distàncies de fins a 50m
- Classe 3: Pot transmetre a distàncies de fins a 10m

L'elecció com a mòdul de comunicació Buetooth va ser el model BISMS02BI-01 de Ezurio, un mòdul de classe 1, que ens assegura una cobertura de fins a 100m sense entrebancs. Ens assegura un bon radi de treball, com per exemple en empreses grans o de més d'una planta. També és un model que ens permet comunicar-nos amb un microcontrolador a través de port sèrie UART, estàndard de comunicació RS-232, que és el que buscàvem per comunicar-nos amb el robot i on a més podem tractar les dades rebudes. La configuració de l'integrat es realitza a través del port sèrie amb comandos estàndard AT. Les dimensions del mòdul Buetooth són les que veiem en la figura 11.4.7.

La comunicació física amb el microcontrolador és realitza a través del port sèrie UART que té el dispositiu, així que només necessitem 2 pins, un per transmetre dades des del Buetooth i un altre per rebre des del microcontrolador.

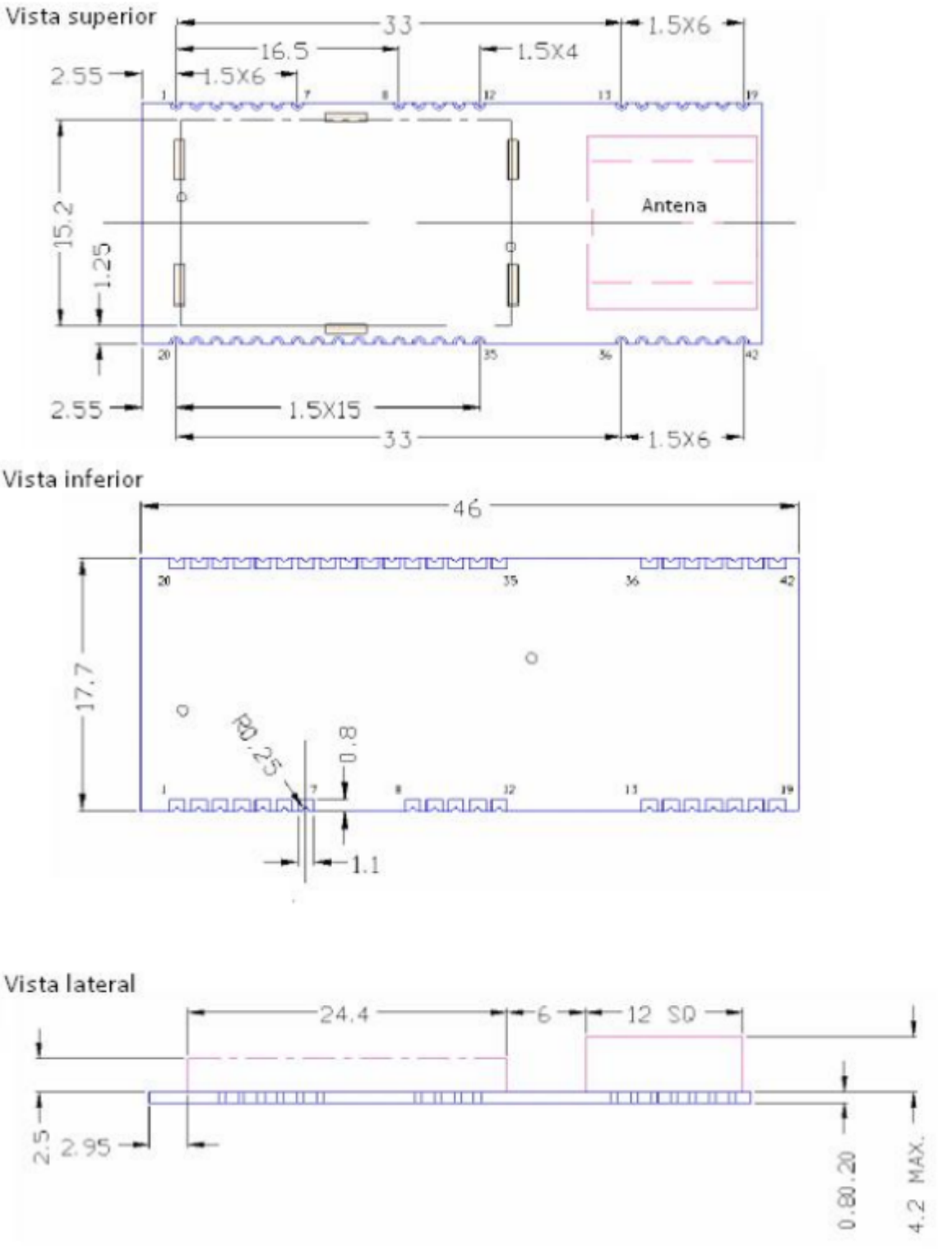


Figura 11.4.7: Dimensions BISMS02BI-01

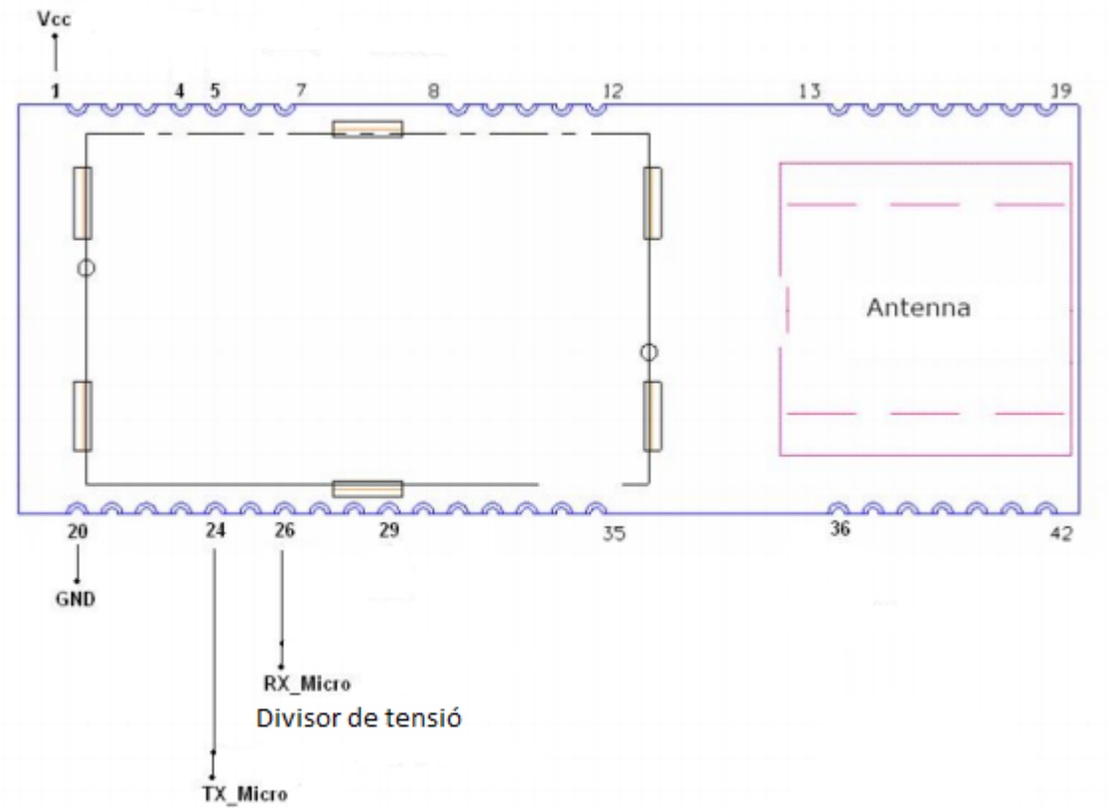


Figura 11.4.8: Pins utilitzats BISMS02BI-01

```
Transmetre dades : Pin 26 (UART_Tx)
Rebre Dades : Pin 24 (UART_Rx)
```

La alimentació del mòdul es realitza amb una tensió de 5V.

```
5V : Pin 1 (Vcc)
GND : Pins 4, 20, 29 (GND)
```

Tenint els pins del mòdul necessaris per utilitzar amb el robot és dissenya el circuit de connexió hardware del Buetooth.

Els pins utilitzats els veiem en la figura 11.4.8.

Per programar la configuració amb que treballarem amb el mòdul Buetooth tenim que connectar els pins del port sèrie TX i RX. Un cop tenim la placa del mòdul Buetooth preparada seguirem els següents passos per la configuració:

- Connectarem el TX del mòdul amb el RX del microcontrolador i el RX del mòdul amb el TX del microcontrolador. Per comunicar-nos amb el mòdul utilitzarem el microcontrolador com a programador Buetooth, on crearem una funció que tot el que enviem des del port sèrie del ordenador anirà directament al mòdul Buetooth.

```
if (usb_serial_available())
        {

                uart_putchar(usb_serial_getchar());

        }

        if (uart_available())
        {

                usb_serial_putchar(uart_getchar());

        }
```

- Obrim l'aplicació Advanced Serial Port Terminal en l'ordenador per poder crear un port COM i introduir les instruccions AT corresponents per la configuració del mòdul.
- Introduïm les següents instruccions AT de configuració seguint l'ordre indicat. Després de cada instrucció el mòdul respon un OK confirmant que la instrucció s'ha executat correctament, ho visualitzarem en el terminal que estem escrivint les instruccions.

```
AT&F* -> Posem tots els registres interns per defecte
AT+BTN=\"Dr Gero\" -> Nom del mòdul
AT+BTK=\"****\" -> Pin del mòdul
ATS512=4 -> Activem la visibilitat i connexió
ATS0=1 -> Configura la auto resposta perquè sigui automàtic
ATE0 -> Eliminem l'echo
AT+BTP -> Activem l'escaneig i esperem una connexió
AT&W -> Guardem en memòria els registres modificats
```

- Ja tenim el mòdul Buetooth configurat

A partir de la primera configuració ja no caldrà configurar el mòdul Buetooth a través de cable. Iniciarem la configuració des del mateix microcontrolador cada vegada que iniciem el robot, així si per alguna raó es des-configura al iniciar-se el tornarem a tenir com volem.

Així doncs, cada vegada que iniciem el robot farem una crida a la funció:

```
bt_config();
```

11.4.1.8. Sensors d'inèrcia, càlcul de l'angle

En aquest apartat determinarem la inclinació del robot basant-nos amb l'acceleròmetre i el giroscopi. Com hem comentat anteriorment, utilitzarem un acceleròmetre tipus MEMS, que tenen alguns avantatges per aquesta aplicació, com un bon ample de banda, bastanta resolució, poc pes, robust i de baix cost. El problema d'aquest tipus de sensors està en què les mesures estan afectades per les acceleracions externes, i la única que ens interessa és l'acceleració de la gravetat.

L'altre sensor que utilitzem, el giroscopi, que ens calcula la velocitat de rotació. Per la mateixa raó que amb el sensor anterior, utilitzem el dispositiu amb tecnologia MEMS. Aquests sensors són immunes a les vibracions però tenen alguna deriva en el temps degut a la integració dels errors del nostre offset.

Així doncs, de forma intuïtiva ens sorgeix la idea de utilitzar la mesura obtinguda per el giroscopi i realitzar la correcció de la deriva amb la mesura de l'acceleròmetre, però la mesura d'aquest últim amb menys valor que l'anterior.

- **Giroscopi**

Primerament farem els càlculs a partir del giroscopi, aquesta mesura, calculada anteriorment a partir del ADC només la passem a rad/s, ja que ens es més còmode treballar . Per tant:

$$gyro_x = gyro_x \cdot \left(\frac{\sqcap}{180}\right)$$

Com hem dit anteriorment, el giroscopi ens dóna les dades de la velocitat en la que estem rotant (velocitat angular). Per aconseguir l'actitud és necessari integrar la velocitat angular, tenint com a concepte d'integració, el multiplicar aquesta dada per un valor determinat "dt".

La constant de temps que utilitzem depèn únicament de les necessitats que tenim, 10ms, que és la duració de la nostra interrupció principal. Sabem també que si els intervals de temps són molt petits, tindrem la necessitat de utilitzar dispositius amb una velocitat molt alta de processament.

Així doncs:

$$\frac{d(ang_x)}{dt} = gyro_x = \text{senyal del giroscopi} = \text{velocitat angular}$$

$$d(ang_x) = gyro_x \cdot dt$$

$$(int)\, velocitatAngular = (int)\, senyalDelGiroscopi = angle$$

Considerant la fòrmules anteriors apreciem que una petita variació de l'angle es pot calcular a partir del producte de la velocitat angular pel temps en el qual s'està realitzant el moviment.

Ara tenim que analitzar, que mitjançant aquestes formules només trobem l'angle en un instant determinat, però aquest no es el nostre objectiu, sinó trobar l'actitud.

Per trobar aquesta actitud, és necessari tenir en compte que el procés descrit abans es realitza en intervals de temps que estableix dt, amb la conseqüència que l'actitud seria el resultat d'un algoritme acumulatiu. Així doncs, obtenim:

$$ang_x(n) = ang_x(n-1) + (gyro_x \cdot dt)$$

On:

$gyro_x$ =senyal del giroscopi

dt =temps de mostreig

$ang_x(n)$ =estat present

$ang_x(n-1)$ =estat anterior

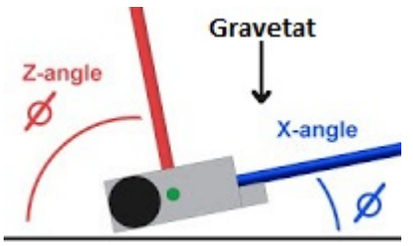


Figura 11.4.9: Eixos acceleròmetre

Però ens trobem amb dos inconvenients principals quan intentem trobar l'actitud només amb el giroscopi, el primer és la deriva, que es produeix al llarg del temps degut al algorisme acumulatiu, és a dir, es tindran valors més fiables de l'actitud només en intervals curts de temps, i en intervals llargs de temps els valors que obtindrem estaran molt lluny de la mesura real.

El segon inconvenient utilitzant el giroscopi és que no podem saber l'estat inicial de l'angle en què comencem, a no ser que tinguem sempre un estat inicial conegut amb el qual podem començar l'algorisme acumulatiu.

Amb tot això, arribem a la conclusió que és impossible aconseguir una mesura fiable de l'actitud només utilitzant el giroscopi, per tant, és necessari la utilització d'un altre dispositiu que ens ajudi amb el problemes del giroscopi. Per això utilitzarem un acceleròmetre.

- Acceleròmetre

Com sabem, el acceleròmetre ens dóna la mesura de l'acceleració a la qual està sotmès el sensor a cada un dels seus eixos, però la seva aplicació més popular és com a sensor d'inclinació.

Hem de tenir en compte que la sensibilitat dels acceleròmetres en el càlcul de la inclinació augmenta quan els seus eixos es troben perpendiculars a la acceleració de la gravetat, és a dir, paral·lels a la superfície de la terra.

Primer em de col·locar l'acceleròmetre a una plataforma del robot, així quan el robot estigui paral·lel a la superfície de la terra, per tant l'eix X i el Z (Per la col·locació del acceleròmetre he d'utilitzar aquests dos eixos) estaran perpendiculars a la gravetat. Com podem observar en la figura 11.4.9:

Com em dit, l'acceleròmetre ens calcularà l'acceleració de la gravetat, és a dir, les forces g's. Sabem que la gravetat accelera els objectes en 1g. Per exemple, si el robot es mou endavant a 1g, el nostre sensor ens marca 2g. Per la majoria de usos en la robòtica els sensors de 2g seran suficients, ja que, si el rang és baix també serà més sensible als canvis de moviment. Però s'ha de tenir en compte que com més baix sigui el rang de l'acceleròmetre més afectat es veurà per les interferències de les vibracions.

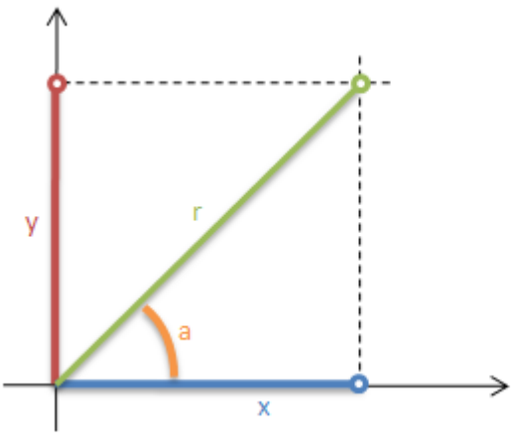


Figura 11.4.10: Per passar de coordenades rectangulars a polars

Així doncs, per calcular l'angle només em d'utilitzar una petita fórmula trigonomètrica (Figura 11.4.10):

$$angle_a = atan(accx_g, -accz_g)$$

On:

$angle_a$ =angle càlculat (a en la figura 11.4.10)

$accx_g$ =g's de l'eix x

$accz_g$ =g's de l'eix z (en la figura 11.4.10 veiem que és l'eix y)

Utilitzem la funció atan2 ja que les dades que ens retorna l'acceleròmetre són cartesianes, però en el cas del projecte treballem amb coordenades polars.

Si bé l'acceleròmetre treballant com a sensor d'inclinació és una opció excel·lent, hem de tenir en compte que aquest sensor ens entrega un angle acertat només en condicions en les quals no s'estigui produint un moviment que impliqui una acceleració lineal a qualsevol dels seus eixos, és a dir, en condicions en què l'acceleració sigui 0. Tenint en compte això, un lleuger canvi en l'acceleració diferent a 0 sobre el sensor o simplement per la presència de vibracions produirà que l'angle que obtenim no sigui l'acertat, amb el qual el sensor continuarà donant informació errònia mentrestant estigui sota aquests efectes.

- Giroscopi i acceleròmetre

Si agafem com a referència els dos mètodes anteriors apreciem clarament que al utilitzar tant el giroscopi com l'acceleròmetre de manera individual, es tenen varis inconvenients que produeixen que el resultat de l'actitud tingui tendència a ser erroni.

Degut a aquest inconvenients ens sorgeix la necessitat de treballar amb els dos sensors simultàniament, amb la finalitat de poder agafar les millors característiques de cada un i que a la vegada contrarestin una mica els inconvenients.

Existeixen alguns mètodes per aconseguir aquesta fusió per atenuar aquests problemes individuals, però nosaltres utilitzarem un filtre complementari.

Aquests filtres són senzills de tractar matemàticament i com tenen una baixa complexitat d'implementació consumeixen pocs recursos. La idea bàsica del filtre és combinar la sortida de l'acceleròmetre i la del giroscopi per obtenir una bona estimació de l'angle, compensant la deriva del giroscopi amb la dinàmica lenta de l'acceleròmetre.

Per crear aquest filtre crearem un algoritme el qual realitza un procés de predicció i un altre de correcció mitjançant la mesura i observació d'un grup de variables del sistema.

Així doncs, la nostra idea principal és agafar l'angle mesurat a partir del giroscopi i del acceleròmetre i veure la diferència:

$$diferència = angle_a - ang_x$$

Un cop tenim aquesta diferència tenim dos opcions, la primera és mirar si la diferència és 0. En aquest cas al ser iguals observem que no hi ha hagut cap tipus de variació respecte els problemes per separat dels dos sensors, per tant, retornarem el valor de l'angle.

És en la segona opció on anem si observem que hi ha una diferència entre l'angle de cadascun. Així doncs, si trobem que hi ha una diferència actuarem de la següent manera:

$$angle = ang_x + \left(\frac{diferència}{K}\right)$$

On:

$angle$ =angle resultant

$angle_x$ =angle del giroscopi

$diferència$ =diferència entre angle del giroscopi i angle del acceleròmetre

K =variable observada

La variable K és la variable independent que decidirà quan cas fem a l'angle mesurat pel acceleròmetre, aquesta variable no es calcula automàticament, sinó que haurem d'observar quin és el valor que ens pot anar millor. Per exemple, si tenim:

- $angle_a$ =60
- ang_x =10
- $diferència$ =50
- Entrem a la segona opció ja que tenim diferència ! = 0
- Provem K =95
- $angle$ =10 +$\left(\frac{50}{95}\right)$ =10.5

En l'exemple anterior veiem que ens creiem molt poc a l'acceleròmetre, ja que és impossible que la diferència sigui tan gran. Utilitzant aquest filtre podem assegurar que l'augment de l'angle serà molt progressiu i que farem poc cas dels angles erronis del acceleròmetre, però com sabem que no

s'equivoca sempre doncs això provocarà que la deriva del giroscopi també és vagi controlant poc a poc.

Després d'assajar aquests tres mètodes de mesura de d'inclinació del robot podem afirmar que els dos primers mètodes utilitzant l'acceleròmetre i el giroscopi de manera individual fan impossible que el robot tingui una bona resposta dinàmica. En el primer cas, la influència de acceleracions diferents a la gravetat impedeix el correcte funcionament, en canvi en el segon cas, s'obtenen millors resultats en el comportament però la deriva produïda pel sensor fa que l'error d'inclinació s'incrementi de manera constant en el temps.

Finalment utilitzant aquest filtre complementari podem assegurar una bona estimació de l'angle i una bona resposta dinàmica, que va resultar satisfactòria en el prototip real.

11.4.2. Implementació de l'algoritme PID

El PID (Proporcional Integral Derivatiu)([PID12, CPI11]) és un mecanisme de control per realimentació que calcula la desviació o error entre un valor mesurat i el valor que és vol obtenir, per aplicar una acció correctora que ajusti el procés. L'algorisme ve donat per tres paràmetres: la proporcional, la integral i el derivatiu.

La proporcional determina l'error entre la posició en la que estem i la que tindríem que estar. La integral soluciona l'error a mesura que ens anem allunyant del punt d'equilibri i donarà més força als motors per recuperar la posició. El derivatiu és la velocitat en què ens estem allunyant de la posició que tindríem que estar, és a dir, la velocitat en què està creixent aquest error. La suma d'aquestes tres accions s'utilitza per ajustar al procés via un element de control, que en el cas d'aquest projecte són els motors. Així doncs, si ajustem aquestes tres variables, tindrem un control dissenyat pel nostre procés. En la figura 11.4.11 veiem un diagrama de blocs de com funciona l'algorisme PID, on en el bloc "control de l'equació" executarem el nostre algorisme.

Per tant, el sistema de control que utilitzarem en aquest projecte és el PID.

En el nostre cas, el valor que volem obtenir és l'angle en el que volem equilibrar el sistema, que serà de 0^{o}, i el valor mesurat serà el valor de l'angle del robot en un instant de temps determinat.

Com ja em dit, l'algorisme de càlcul de control PID és dóna a partir de tres paràmetres: la proporcional, la integral i el derivatiu.

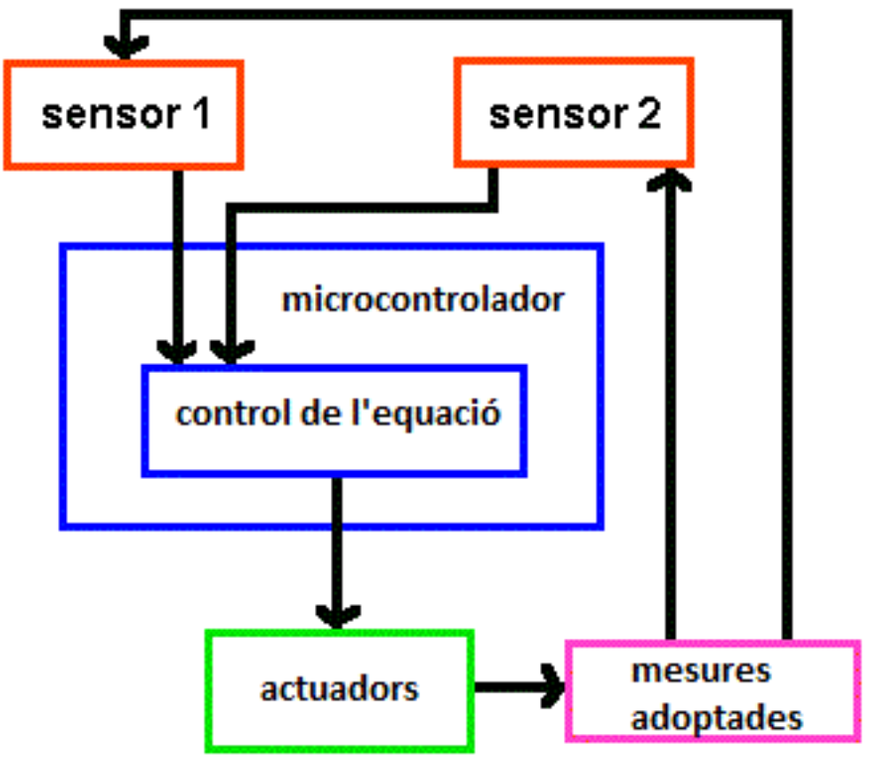


Figura 11.4.11: Diagrama de blocs algorisme PID

- La proporcional

La part proporcional consisteix en el producte entre la senyal d'error i la constant proporcional. La nostra meta és que l'estat continuat d'error sigui casi nul. Existeix per això un valor límit en la constant proporcional , ja que sinó es crearà el fenomen conegut com sobre-oscil·lació, i és recomanable que aquesta part no produeixi aquest fenomen. La nostra intenció a l'hora de considerar la proporcional és que el robot es mantingui als 0º però la part proporcional no considera el temps, per tant la millor manera de solucionar l'error permanent i fer que el sistema tingui alguna component que tingui en compte la variació respecte el temps, és incloent les accions integral i derivada.

En resum, el valor proporcional determina la reacció de l'error actual.

- La integrada

El mode de control integral té com a propòsit disminuir l'estat continuat d'error, provocat pel mode proporcional. El control integral actua quan hi ha una desviació entre la variable i el punt desitjat, integrant aquesta desviació en el temps i sumant-la a l'acció proporcional. Després, la resposta integral es sumada a la proporcional, per formar el control P + I, amb el propòsit d'obtenir una resposta estable del robot sense error constant.

El control integral s'utilitza per obviar l'inconvenient de la desviació permanent de la variable respecte el punt desitjat.

En resum, aquest mode integral genera una correcció proporcional a l'estat continuat d'error, això ens assegura que aplicant un esforç de control suficient, aquest error continuat es redueix a 0.

El valor de la integral el limitarem entre -0.35 i 0.35, ja que si no el limitem podem tenir una sobre-oscil·lació generada per l'estat continuat d'error.

$$var_I = flim\left(var_I, -0{,}35, 0{,}35\right);$$

On flim només ens limita el valor que li enviem (en aquest cas, var_I) entre els valor de -0.35 i 0.35.

- La derivada

La acció derivativa es torna visible quan hi ha un canvi en el valor absolut de l'error, si aquest error és constant aquesta acció no actua. Aquest error és la desviació existent entre el valor mesurat i el valor desitjat. La funció d'aquesta acció és mantenir l'error al mínim corregint-lo a la mateixa velocitat que s'està produint, d'aquesta manera evitarem que l'error s'incrementi.

És a dir, la derivada determina la reacció en el temps en el que l'error es produeix.

La sortida d'aquests tres termes són sumats per calcular la sortida del controlador PID. Definint "u" com a sortida del controlador, quedant de la següent manera:

$$u = -K_p e - K_i \int_0^t e dt - K_d \frac{de}{dt}$$

Aquesta fórmula a l'hora de passar-la a la pràctica no es pot utilitzar tal qual en el codi, per tant convertirem la integral en un sumatori i la derivada en una variació de l'error. El resultat, és la velocitat en què el motors tindran que donar voltes a les rodes en cada moment per mantenir-se en equilibri.

Així doncs, la fórmula utilitzada és aquesta:

$$u = (ang_x \cdot K_p) + (gyro_x \cdot K_d) + (var_i \cdot K_i)$$

On:

ang_x =és l'angle en què ens trobem

K_p =constant proporcional

$gyro_x$ =és la velocitat angular

K_d =constant derivada

var_i =sumatori de l'error continuat

K_i =constant integrada

La tècnica utilitzada per trobar les constants correctes ha estat la de assaig/error, fins a trobar les constants adients per a que el robot estigues estable.

Per poder implementar el PID necessitarem almenys tres elements:

1. Un sensor amb el que podrem obtenir el valor de l'angle del robot periòdicament, que en el nostre cas seran dos, l'acceleròmetre i el giroscopi.
2. Un controlador en el que s'implementi el PID per transformar la senyal d'error entre l'angle d'equilibri i l'angle dels sensors a un valor per enviar als actuadors, en el nostre cas és el Atmega32u4.
3. Uns actuadors amb els que provocarem un canvi en el moviment del sistema amb la finalitat de corregir l'angle del robot, en el nostre cas són el motors i les rodes.

Els sensors proporcionen una senyal al controlador, la qual representa el punt actual en el que es troba el robot. El controlador llegeix una senyal externa que representa el valor al que volem arribar. El controlador obté la senyal d'error, que determina a cada instant la diferència que hi ha entre el valor desitjat i el valor mesurat. La senyal d'error és utilitzat per cada un dels 3 components del controlador PID. Les tres senyals sumades, composen la senyal de sortida que el controlador utilitzarà per moure l'actuador. La senyal resultant de la suma de les 3 components és la variable manipulada i no s'aplica directament als motors, sinó que ha de ser transformada per ser compatible amb l'actuador que utilitzem.

El pes de la influència que cada una de les tres parts té a la suma final ve dónat per la constant proporcional, el temps integral i el temps derivatiu. Pretenem així que la nostra interrupció principal de control solucioni eficaçment i en el mínim temps possible les pertorbacions del robot, per així poder mantenir-se estable.

Capítol 12

Manual usuari aplicació Java enllaç robot

Per gestionar el robot he creat una aplicació amb Java. Aquesta aplicació ha estat creada específicament per gestionar i controlar el robot. Aquesta aplicació es pot executar des de qualsevol sistema Windows de 32 i 64 bits. Veiem una captura del menú principal en la imatge 12.0.1. On veiem que em separat les diferents funcions:

- **Connect:** Aquesta caixa esta formada per 2 parts, un botó i una llista desplegable. El botó està format per tres opcions:
  - **Check:** Fa una búsqueda dels ports sèrie disponibles, un cop ha trobat els port sèries ens surt la següent opció. Imatge 12.0.2.
  - **Conectar:** En aquesta opció desplegarem la llista desplegable i seleccionarem el port per fer la connexió amb el nostre robot. Un cop s'hagi establert la connexió el botó canviarà a la següent opció. Imatge 12.0.3.
  - **Desconectar:** Aquesta opció serveix per desconnectar la connexió. Imatge 12.0.4.

- **Documents:** La caixa Documents ens indica si tenim algun document a la "bústia" del robot, indicant-nos amb color verd si és troba el document o amb vermell si no es troba. Imatge 12.0.5.

- **PID:** En aquesta caixa podem modificar la configuració del PID, i canviar les constants de KD, KI i KP, també podem canviar el valor del offset del acceleròmetre (molt útil en cas que tinguem alguna petita pendent en el terreny). Imatge 12.0.6.

- **Info:** La finalitat d'aquesta caixa només és informativa, ens dóna la telemetria d'alguns valor importants del robot. Imatge 12.0.7.
  - **Angle:** angle que hem calculat.
  - **Giroscopi:** valor de la velocitat angular.

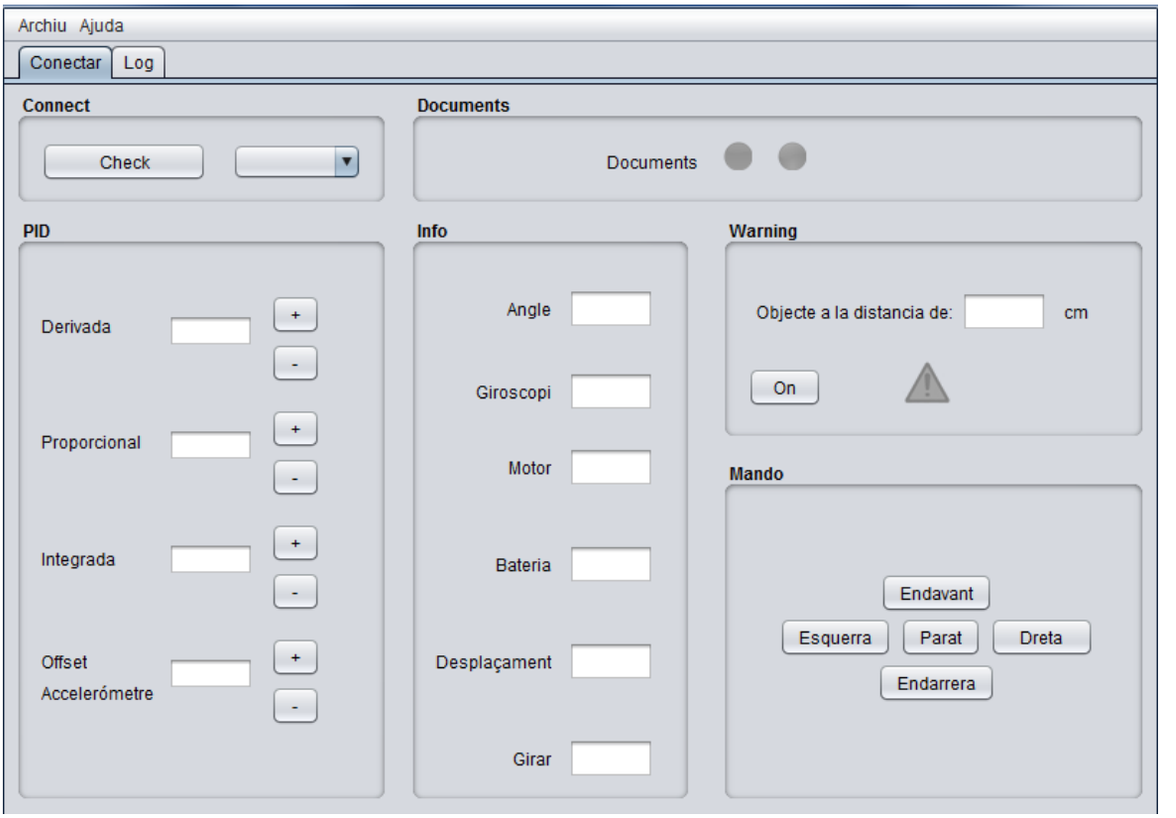


Figura 12.0.1: Menú principal

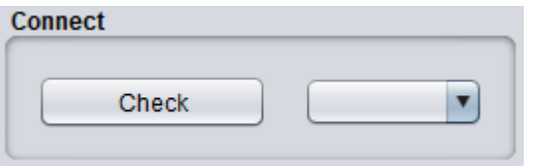


Figura 12.0.2: Opció check

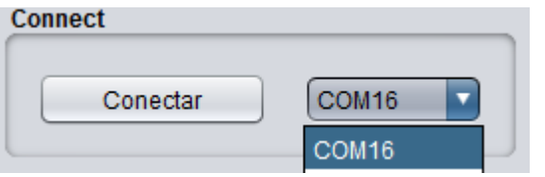


Figura 12.0.3: Opció conectar

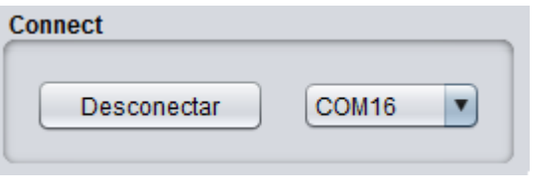


Figura 12.0.4: Opció desconectar

Figura 12.0.5: Caixa documents

Figura 12.0.6: Caixa PID

Figura 12.0.7: Caixa info

- **Motor:** valor que enviem al motor.
- **Bateria:** voltatge que tenim a la bateria.
- **Desplaçament:** en cas d'estar en moviment, valor en què és mou endavant o endarrere.
- **Girar:** valor de gir, cap a l'esquerra o dreta.

- **Warning:** Aquesta caixa ens avisa si estem apunt de topar amb un objecte, ens informa de la distància on es troba i a més ens permet desactivar el moltes vegades molest Buzzer amb un botó on/off. Imatge 12.0.8.

- **Mando:** Caixa des de on controlem el robot des de el ratolí amb les opcions de endavant, endarrere, dreta, esquerra i parat. Imatge 12.0.9. També tenim la possibilitat de controlar el robot a partir de les fletxes del téclat, només em de tenir l'aplicació oberta.

  - **Fletxa amunt:** endavant
  - **Fletxa avall:** endarrere
  - **Fletxa dreta:** dreta
  - **Fletxa esquerra:** esquerra

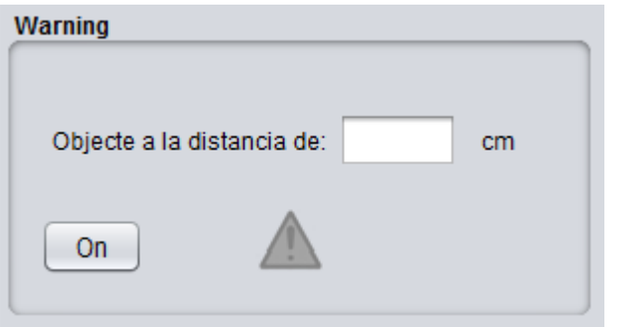


Figura 12.0.8: Caixa warning

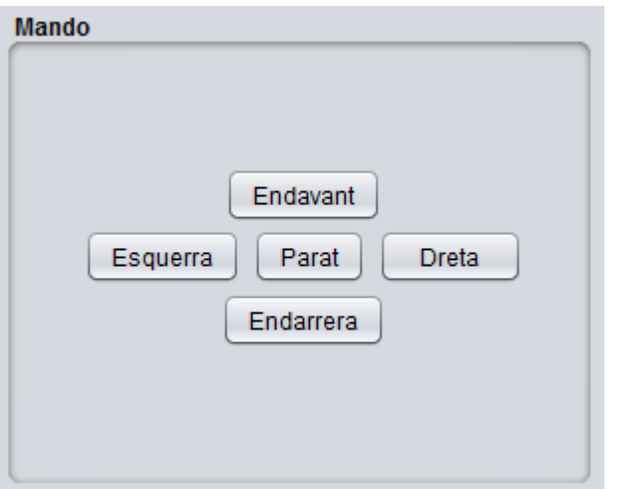


Figura 12.0.9: Caixa mando

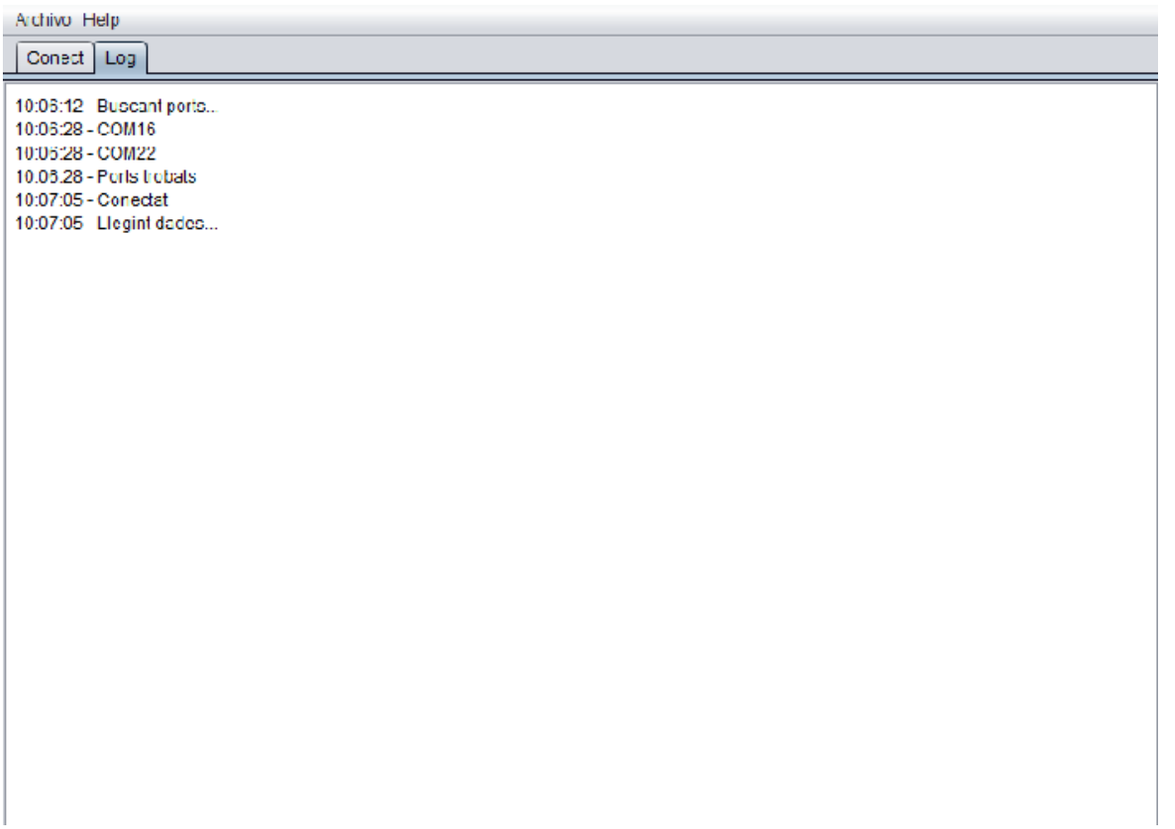


Figura 12.0.10: Pestanya log

- **Tecla espai:** parat

A part de les caixes de control i informatives tenim també una pestanya de log, on podem accedir per monitoritzar tot el que em fet en aquella sessió. Imatge 12.0.10.

Al menú principal tenim:

- **Archiu** (Imatge 12.0.11)
  - **Exit:** per sortir del programa
- **Ajuda** (Imatge 12.0.12)

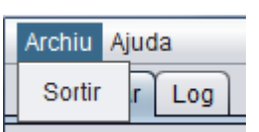


Figura 12.0.11: Menú principal - Opció Archiu

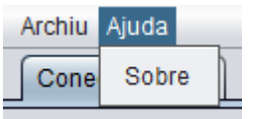


Figura 12.0.12: Menú principal - Opció Ajuda

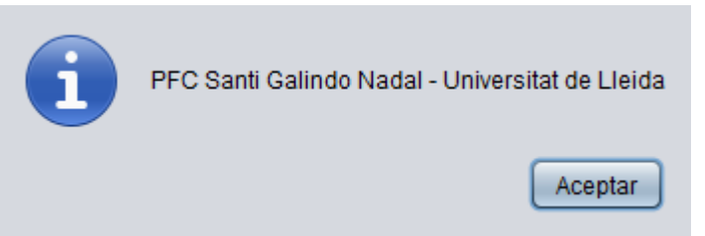


Figura 12.0.13: Menú principal - Opció Sobre

- **Sobre:** informació del autor. Imatge 12.0.13)

Capítol 13

Resultats

13.1. Reaccions del robot

El sistema, com a resultat final ha aconseguit realitzar l'equilibri. A mesura que la funcionalitat del PID progressava anava mostrant clares millores. El problema principal em va sorgir a l'hora de ajustar el PID, ja que no vaig poder implementar un sistema per poder observar en temps real les reaccions i millores més que pel mètode de assaig/error.

El mètode seguit a l'hora de buscar una bona configuració per aconseguir l'estabilitat correcta va ser configurar el PID per separat, en el següent ordre:

- **P**

El primer valor a establir va ser la proporcional, on un cop teníem el valor de l'angle calculat ja podíem utilitzar-lo per aguantar l'equilibri en unes condicions del terreny gairebé perfectes i sense cap tipus de moviment extern.

Per ajustar la proporcional els passos a seguir van ser senzills. Donar valors a KP fins que el robot sobre-oscil·lava, un cop passava això baixàvem el valor aproximadament un 20 %.

El valor final de la KP va ser 13.5, on només amb la següent fórmula aconseguíem aguantar l'equilibri en condicions de l'angle bastant propers a 0^{o}, però amb el problema que no recuperava a mesura que s'inclinava.

$$motor = (ang_x \cdot K_p)$$

- **P+D**

Un cop vam tenir la proporcional el següent pas va ser establir el valor de la derivada. Configurant aquest valor el primer que s'observa és una millora quan tenim un canvi brusc de l'angle, fent que la reacció fos ràpida en el cas que el robot s'inclinés a una velocitat lleugerament forta.

Després de varies proves, el valor final de KD va ser 1.4, on s'aconseguia una equilibri entre KP i KD, tenint així una bona reacció en canvis d'angle bastant ràpids i un control bastant bo a l'hora de mantenir-se a prop dels 0^{o}.

Per configurar KD, els passos a seguir van ser els mateixos que per KP. Pujar el valor de KD fins que observàvem una sobre-oscil·lació, un cop això passava ajustàvem els dos valors per aconseguir aquest equilibri. Per exemple, amb un valor de KD de 1.6 el robot ja creava aquesta sobre-oscil·lació amb una petita velocitat angular.

Així doncs, amb la següent fórmula ja aconseguim l'equilibri desitjat.

$$motor = (ang_x \cdot K_p) + (gyro_x \cdot K_d)$$

Però s'observava que amb un angle petit la resposta del robot no era adequada, ja que arribava a trobar una situació d'homeòstasi amb un angle petit i constant (sense velocitat angular) on la resposta dels motors no era suficient per contrarestar l'error de l'angle.

- **P+D+I**

Aquesta va ser la part més complicada d'ajustar, ja que pel mètode de assaig/error es difícil de trobar una bona configuració.

Així doncs, per solucionar l'efecte anterior, crearem una nova variable que s'incrementarà progressivament a mesura que passi el temps, arribant el punt on serà prou important dintre del PID com per contrarestar l'efecte que no pot solucionar PD.

$$motor = (ang_x \cdot K_p) + (gyro_x \cdot K_d) + (var_i \cdot K_i)$$

Després de varies proves, el valor de KI final va ser 9. Fent que el robot aguantés l'equilibri i que els efectes externs o d'acumulació en el temps no l'afectessin en gran mesura.

Un cop aconseguit que el robot mantingués l'equilibri el següent pas va ser detectar objectes a partir del sensor d'ultrasons. El factor principal era decidir la distància en què detectava l'objecte per poder parar, en cas d'estar en moviment, i sense col·lisionar amb l'objecte. La distància aquesta, fent proves d'assaig/error, va ser 65cm, on el robot tenia temps de detectar i parar sense col·lisionar amb l'objecte.

El resultat negatiu del projecte va ser a l'hora de utilitzar el robot per seguir una línia, on la detecció i gir del robot era bona però es creava una inestabilitat que feia que el robot es separés de la línia. En canvi, si érem nosaltres els que aguantàvem l'equilibri del robot el seguiment de la línia era bo. Vaig decidir prescindir del seguidor de línies, ja que per mi era molt més prioritari que aguantés l'equilibri.

Un altre resultat inesperat va ser amb el control remot del robot, on el robot interpretava aquest desplaçament com un error i volia recuperar-la, fent que el moviment fos molt dolent. Va ser en aquest moment quan la necessitat d'uns encoders es va fer molt gran, ja que així hagués pogut fer un altre PID pels valors dels encoders i donar-li un desplaçament a partir del moviment de les rodes. La solució va ser donar-li aquest desplaçament a partir del offset de l'angle, on l'hi dèiem que l'angle bo era un grau diferent a 0.

El mateix va passar per girar, ja que al donar-li un valor més gran a una roda aquesta s'accelerava buscant l'equilibri i provocava que el robot no aguantés, amb la conseqüència que el robot queia, la manera de solucionar això era que el gir fos molt petit.

Tot i no tenir encoders, els resultats han sigut molt bons, dónant una estabilitat tan en parat com en moviment i detectant objectes amb bastanta precisió. També estic molt satisfet amb la comunicació per Buetooth, que em permet rebre i enviar dades amb una gran velocitat.

Així doncs, podem dir que em complert els objectius inicials principals, amb l'excepció de seguir una línia.

13.2. Imatges del muntatge

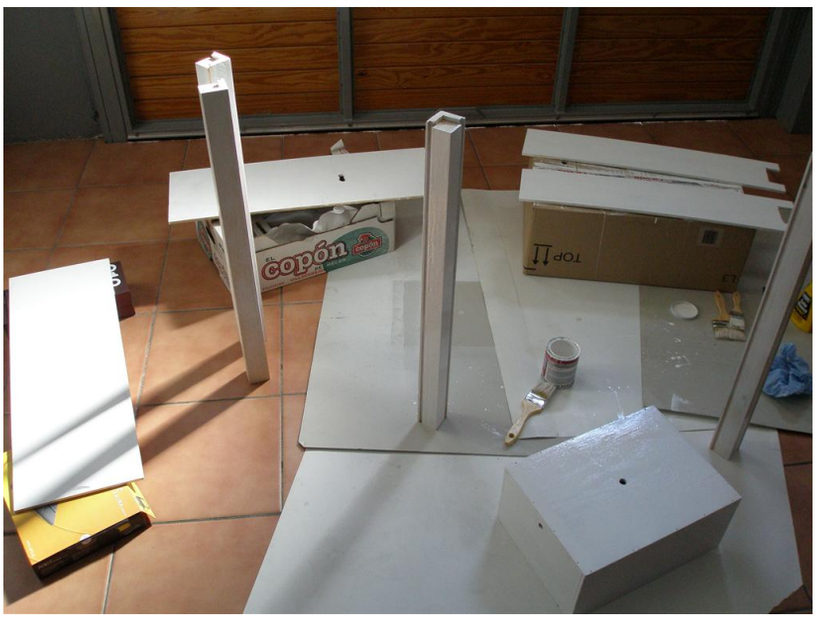


Figura 13.2.1: Peces pintades

Figura 13.2.2: Peces pintades 2

Figura 13.2.3: Peces pintades 3

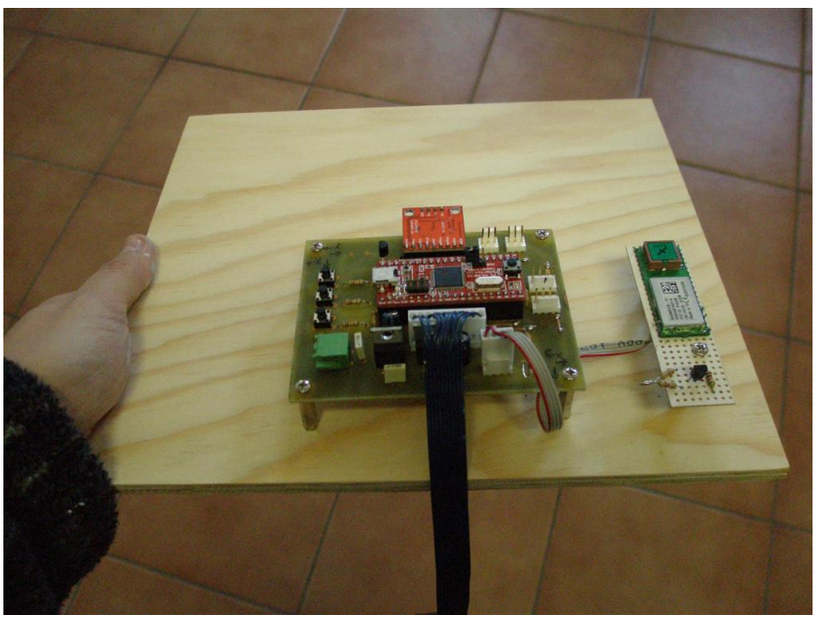


Figura 13.2.4: Placa Soldada

Figura 13.2.5: Motors i Ponts H

Figura 13.2.6: Carcasa

Figura 13.2.7: Carcasa amb placa

Figura 13.2.8: Vista motors i pont H

Figura 13.2.9: Carcasa amb placa i rodes

Figura 13.2.10: Carcasa final

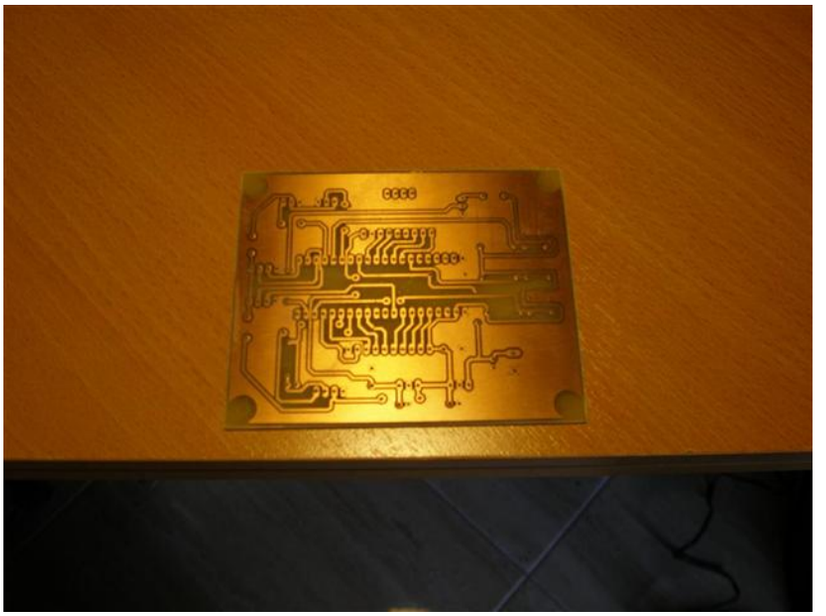


Figura 13.2.11: Placa sense soldar

Capítol 14

Conclusions finals

14.1. Conclusions

Després de tot el comentat en capítols anteriors, ja només ens queda extreure conclusions de tot el treball realitzat durant al llarg d'aquest projecte.

Un cop finalitzat el projecte i fent un anàlisis d'on hem arribat i els objectius proposats, podem dir que hem acomplert els objectius principals, a excepció de l'objectiu d'aconseguir que el robot seguís una línia, però que no ha afectat en cap moment al funcionament principal del robot. A més s'ha exposat un algorisme matemàtic capaç de fusionar les senyals del giroscopi i l'acceleròmetre, anomenat filtre PID.

La búsqueda de les variables a ser utilitzades pel filtre PID requereix un ampli estudi matemàtic, ja que el fet d'implementar-les pel mètode assaig/error provoca que aconseguir una correcta calibració del filtre sigui una tasca difícil.

La utilització correcta de la senyal del giroscopi presenta un greu problema degut especialment a la deriva, pròpia d'aquests tipus de sensors i al no poder eliminar-se per complet, implica que un giroscopi per aquesta aplicació no es pugui utilitzar sol.

L'acceleròmetre és una bona solució per aplicacions on es necessita tenir una mesura d'inclinació, sempre i quan el lloc on l'instal·lem no estigui sotmès a vibracions significatives o a moviments que provoquin acceleracions.

El dubte que se'ns presenta al parlar de giroscopis i acceleròmetres per aplicacions d'inclinació és quin dels dos utilitzar. Realment els dos són capaços de mesurar la inclinació, però aquesta no és la seva veritable naturalesa d'ús, ja que l'acceleròmetre mesura acceleracions i el giroscopi la velocitat angular. Veient això cadascun té les seves respectives avantatges sobre l'altre, mentre que l'acceleròmetre té problemes davant les vibracions i acceleracions, el giroscopi no ho és i mentre el giroscopi té una deriva en la seva mesura, l'acceleròmetre no. Així doncs, la unió del acceleròmetre i el giroscopi ajuda a millorar els problemes dels dos sensors, una unió encertada.

Tenia molts dubtes sobre la comunicació via Buetooth, ja que no esperava que la comunicació fos tan estable ni que la distància fos tan gran, a més, al permetre'ns comunicar-nos amb qualsevol dispositiu que accepti protocol sèrie, ens permet fer un software per comunicar-nos.

Personalment he disfrutat molt de l'experiència de crear un robot des de 0, el qual m'ha portat a posar a la pràctica molts coneixement adquirits al llarg del estudis cursats, no obstant, s'ha hagut

d'aprofunditzar en diferents camps, tant en electrònica com en programació, cosa que m'ha servit per augmentar els coneixements d'aquests estudis. Ha sigut necessari l'estudi i la reflexió en profunditat de tots els elements que composen un robot mòbil per poder comprendre les necessitats que aquests puguin tenir, en especial el funcionament del microcontrolador Atmega32u4, el mòdul IMU i els diferents algorismes com el PID o el del port sèrie. Abans de començar va ser necessari documentar-se sobre el funcionament i llenguatge utilitzat per les diferents eines de desenvolupament de programació de microcontroladors i Java.

14.2. Futures extensions

En aquest apartat realitzarem un anàlisis de les futures extensions que es podrien realitzar a partir d'aquest projecte.

La primera idea que ens ve a la ment després d'haver realitzat el projecte és la de col·locar uns encoders al robot per donar més estabilitat i poder-nos desplaçar d'una manera més senzilla. Per això només faria falta comprar uns motors amb encoders, o simplement adaptar-los als nostres motors. Així doncs, amb uns encoders obtindríem un robot més precís i millor davant de les pertorbacions, a més ens permetria desplaçar-nos i mantenir l'equilibri en una rampa sense problemes.

Una altre factor que es podria millorar és la bateria. Aquest tipus de robots necessiten en tot moment alimentació per fer funcionar el sistema de control, cosa que si el sistema deixa de funcionar a causa de la bateria el robot cau. Això és podria resoldre insertant una pota que funcioni amb un relé. Aquest relé estaria normalment tancat evitant que la pota toqui a terra. Un cop el robot és quedés amb poca energia, el relé s'obriria i la pota faria que el robot no caigués a terra.

Un altre paràmetre important pel funcionament del robot, seria reduir el pes. Una reducció del pes ens ajudaria a tenir un millor control i ens permetria transportar el robot sense tants problemes. Per realitzar aquesta proposta es tindria que redissenyar el robot o canviar els materials del xasis extern del robot, com per exemple alumini, que no és tant pesat com la fusta.

També s'ha pensat en inserir una càmera i un mòdul wireless per poder controlar el robot a una distància més gran. Per fer aquest canvi només tindríem que buscar un mòdul wireless que suportés protocol sèrie i la càmera que portés integrat wireless per enviar les imatges i fins i tot que dotés al robot de la capacitat de reconèixer alguns objectes.

Una altra idea futura és instal·lar un sistema de navegació, això ens ajudaria a que el transport fos més automàtic.

En el tema comunicació, es podria fer una aplicació per dispositiu mòbil o tablet per la comunicació, així com el control i la configuració del robot, que ens donaria més mobilitat.

Per últim, s'ha pensat en aprofitar el balanceig natural del sistema. Es podria dissenyar un control que permeti que el robot es balancegi suaument d'una banda a l'altra, com per exemple, un control repetitiu. Com per exemple podria servir per un bressol de nen petit.

Algunes altres aplicacions podrien ser un carro de la compra, una cadira per minusvalidesa, o com a robot d'àmbit domèstic.

Bibliografia

[23212] Web dedicada a la divulgació publica de coneixement http://es.wikipedia.org/wiki/RS-232 - Visitat el 2011 11.1.2.2

[SIL07] Una panorámica de los robots móviles. R. Silva Ortigoza, J. R. García Sánchez, V. R. Barrientos Sotelo, M. A. Molina Vilchis. Universidad Rafael Belloso Chacín. Méjico, 2007 2

[FON09] . Justin Fong, Simon Uppill. Faculty of engineering, computer and mathematical sciences. University of Adelaide. Australia, 2009. 5

[DAT11] Datasheet's

Bibliografia

[ATM11] Documents resources - www.atmel.com - Visitat el 2011 11.1.2

[PID12] Programming PID - www.societyofrobots.com - Visitat el 2012 11.4.2

[ROB12] http://www.robotmarketplace.com/ - Visitat el 2011 8

[PRE12] www.robotpremiere.com - Visitat el 2011 8

[ADX12] http://www.foroselectronica.es/50/acelerometro-adxl-1477-5.html - Visitat el 2012 10

[PAL08] http://www.neoteo.com/mems-las-nanomaquinas-que-cambiaran-al-mundo - Visitat el 2011 2

[GYR12] http://invensense.com/mems/technology.html - Visitat el 2012 10

[EAG11] http://www.cadsoft.de/wp-content/uploads/2011/05/manual-eng.pdf - Visitat el 2011 6.1.6

[CPI11] http://rjv11.blogspot.com.es/2011/03/practica-4-control-pid-el-robot.html - Visitat el 2012 11.4.2

[RS211] http://arvinformatica.wordpress.com/2011/03/11/manejo-del-puerto-serial-rs232-con-java/ - Visitat el 2011 11.1.2.2

[MTD11] http://www.mattairtech.com/index.php/development-boards/atmega32u4-usb-development-board-arduino-compatible.html - Visitat el 2011 10, 11.1.2

[QTR11] http://www.jmnlab.com/robotzero/rzvd5.html - Visitat el 2011 10

Annex

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 16/32K Bytes of In-System Self-Programmable Flash (ATmega16U4/ATmega32U4)
  - 1.25/2.5K Bytes Internal SRAM (ATmega16U4/ATmega32U4)
  - 512Bytes/1K Bytes Internal EEPROM (ATmega16U4/ATmega32U4)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
    - All supplied parts are preprogramed with a default USB bootloader
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- USB 2.0 Full-speed/Low Speed Device Module with Interrupt on Transfer Completion
  - Complies fully with Universal Serial Bus Specification Rev 2.0
  - Supports data transfer rates up to 12 Mbit/s and 1.5 Mbit/s
  - Endpoint 0 for Control Transfers: up to 64-bytes
  - 6 Programmable Endpoints with IN or Out Directions and with Bulk, Interrupt or Isochronous Transfers
  - Configurable Endpoints size up to 256 bytes in double bank mode
  - Fully independent 832 bytes USB DPRAM for endpoint memory allocation
  - Suspend/Resume Interrupts
  - CPU Reset possible on USB Bus Reset detection
  - 48 MHz from PLL for Full-speed Bus Operation
  - USB Bus Connection/Disconnection on Microcontroller Request
  - Crystal-less operation for Low Speed mode
- Peripheral Features
  - On-chip PLL for USB and High Speed Timer: 32 up to 96 MHz operation
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - Two 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - One 10-bit High-Speed Timer/Counter with PLL (64 MHz) and Compare Mode
  - Four 8-bit PWM Channels
  - Four PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Six PWM Channels for High Speed Operation, with Programmable Resolution from 2 to 11 Bits
  - Output Compare Modulator
  - 12-channels, 10-bit ADC (features Differential Channels with Programmable Gain)
  - Programmable Serial USART with Hardware Flow Control
  - Master/Slave SPI Serial Interface

8-bit AVR® Microcontroller with 16/32K Bytes of ISP Flash and USB Controller

ATmega16U4
ATmega32U4

Preliminary

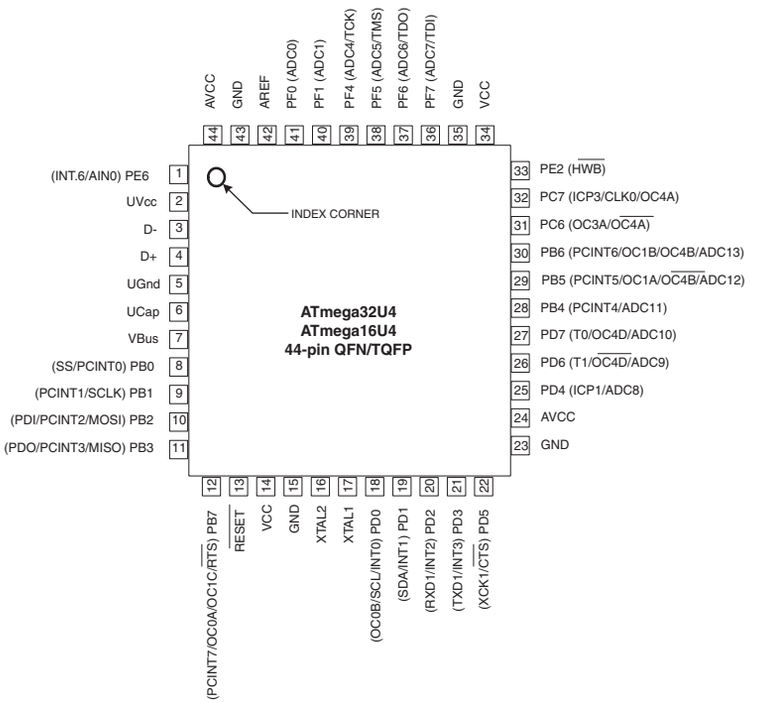
7766F–AVR–11/10

  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
  - On-chip Temperature Sensor
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal 8 MHz Calibrated Oscillator
  - Internal clock prescaler & On-the-fly Clock Switching (Int RC / Ext Osc)
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - All I/O combine CMOS outputs and LVTTL inputs
  - 26 Programmable I/O Lines
  - 44-lead TQFP Package, 10x10mm
  - 44-lead QFN Package, 7x7mm
- Operating Voltages
  - 2.7 - 5.5V
- Operating temperature
  - Industrial (-40°C to +85°C)
- Maximum Frequency
  - 8 MHz at 2.7V - Industrial range
  - 16 MHz at 4.5V - Industrial range

Note: 1. See “Data Retention” on page 8 for details.

1. Pin Configurations

Figure 1-1. Pinout ATmega16U4/ATmega32U4



2. Overview

The ATmega16U4/ATmega32U4 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16U4/ATmega32U4 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

9. Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega16U4/ATmega32U4. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 15.

9.1 Interrupt Vectors in ATmega16U4/ATmega32U4

Table 9-1. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	Reserved	Reserved
7	$000C	Reserved	Reserved
8	$000E	INT6	External Interrupt Request 6
9	$0010	Reserved	Reserved
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	USB General	USB General Interrupt request
12	$0016	USB Endpoint	USB Endpoint Interrupt request
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	Reserved	Reserved
15	$001C	Reserved	Reserved
16	$001E	Reserved	Reserved
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI (STC)	SPI Serial Transfer Complete
26	$0032	USART1 RX	USART1 Rx Complete
27	$0034	USART1 UDRE	USART1 Data Register Empty
28	$0036	USART1TX	USART1 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator

Table 9-1. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address[2]	Source	Interrupt Definition
30	$003A	ADC	ADC Conversion Complete
31	$003C	EE READY	EEPROM Ready
32	$003E	TIMER3 CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3 COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3 COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3 COMPC	Timer/Counter3 Compare Match C
36	$0046	TIMER3 OVF	Timer/Counter3 Overflow
37	$0048	TWI	2-wire Serial Interface
38	$004A	SPM READY	Store Program Memory Ready
39	$004C	TIMER4 COMPA	Timer/Counter4 Compare Match A
40	$004E	TIMER4 COMPB	Timer/Counter4 Compare Match B
41	$0050	TIMER4 COMPD	Timer/Counter4 Compare Match D
42	$0052	TIMER4 OVF	Timer/Counter4 Overflow
43	$0054	TIMER4 FPF	Timer/Counter4 Fault Protection Interrupt

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Memory Programming" on page 346.

2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

Table 9-2 shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 9-2. Reset and Interrupt Vectors Placement[1]

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Note: 1. The Boot Reset Address is shown in Table 27-8 on page 344. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

9.1.1 Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

13. 8-bit Timer/Counter0 with PWM

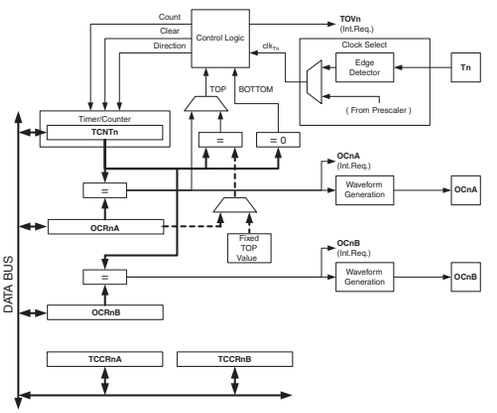
Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **Three Independent Interrupt Sources (TOV0, OCF0A, and OCF0B)**

13.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 13-1. For the actual placement of I/O pins, refer to "Pinout ATmega16U4/ATmega32U4" on page 3. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "8-bit Timer/Counter Register Description" on page 102.

Figure 13-1. 8-bit Timer/Counter Block Diagram



13.1.1 Registers

The Timer/Counter (TCNT0) and Output Compare Registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B). See “Output Compare Unit” on page 93. for details. The Compare Match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

13.1.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in the table below are also used extensively throughout the document.

Table 13-1.

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

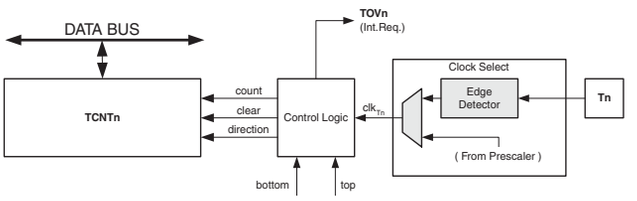
13.2 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler, see “Timer/Counter0, Timer/Counter1, and Timer/Counter3 Prescalers” on page 89.

13.3 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 13-2 shows a block diagram of the counter and its surroundings.

Figure 13-2. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.

direction Select between increment and decrement.

clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 96.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

13.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and Compare Output mode (COM0x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (“Modes of Operation” on page 96).

Figure 13-3 shows a block diagram of the Output Compare unit.

24. Analog to Digital Converter - ADC

24.1 Features

- **10/8-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **± 2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **Twelve Multiplexed Single-Ended Input Channels**
- **One Differential amplifier providing gain of 1x - 10x - 40x - 200x**
- **Temperature sensor**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{CC} ADC Input Voltage Range**
- **Selectable 2.56 V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The ATmega16U4/ATmega32U4 features a 10-bit successive approximation ADC. The ADC is connected to an 12-channel Analog Multiplexer which allows six single-ended voltage inputs constructed from several pins of Port B, D and F. The single-ended voltage inputs refer to 0V (GND).

The device also supports 32 differential voltage input combinations, thanks to a differential amplifier equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 10 dB (10x), 16dB (40x) or 23dB (200x) on the differential input voltage before the A/D conversion. Two differential analog input channels share a common negative terminal (ADC0/ADC1), while any other ADC input can be selected as the positive input terminal. If 1x, 10x or 40x gain is used, 8-bit resolution can be expected. If 200x gain is used, 7-bit resolution can be expected.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 24-1.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ± 0.3V from V_{CC}. See the paragraph “ADC Noise Canceler” on page 301 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

24.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AV_{CC} or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential amplifier.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. The ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

24.3 Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal is still set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event.

MT-DB-U4 User Guide

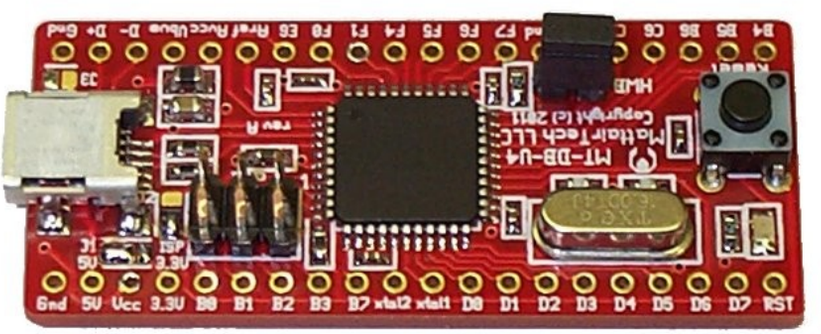


Table of Contents

Overview

Introduction

The MT-DB-U4 is a development board for the Atmel ATmega32U4 USB microcontroller. The board has 40 pins in a dual inline configuration with 100 mil pin spacing and 700 mil row spacing which allows for easy mounting on a breadboard. It includes a mini USB connector, status LED, 16MHz crystal, reset button, HWB boot jumper, and ISP header. A DFU bootloader comes preinstalled which allows programming of the chip over USB without an external programmer. The ISP header can be used with an external programmer for in-system programming. This header can be reconfigured to allow the MT-DB-U4 itself to be an ISP programmer, or to be used as a SPI master or slave. The board can be powered via USB at 5V or it can be externally powered (3V - 5.5V). All programmable IO pins are routed to headers, including those used by on-board hardware. The chip can be clocked externally, and the board is compatible with HV programming. The USB connections are also routed to header pins, which allows for panel-mount USB connectors. The analog power rail is filtered and a separate ground plane lies under the analog traces with a separate analog ground header pin. The PCB is high-quality with ENIG (gold-plated) finish, red soldermask, and white screenprinting showing the pinout. It measures approximately 2.1” x 0.9” (52mm x 23mm) and is 0.062” (1.6mm) thick.

MT-DB-U4 Features

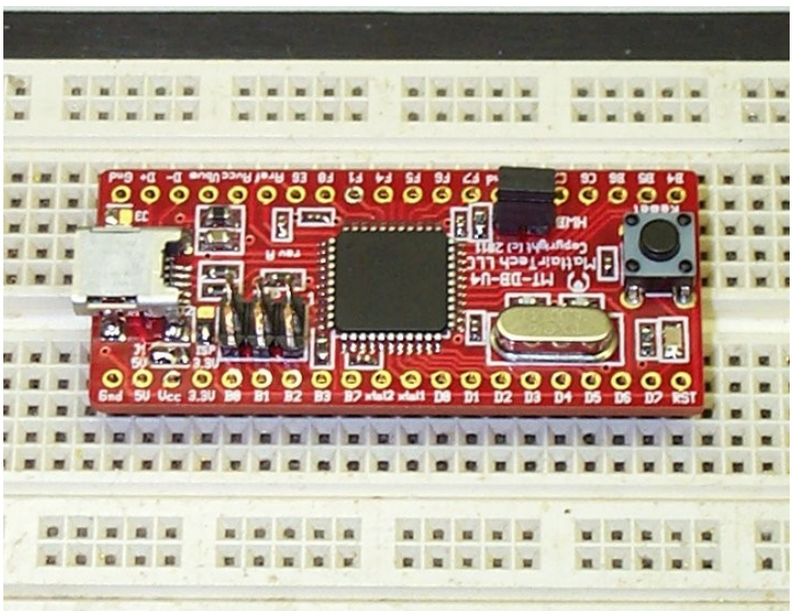
- ATmega32U4 USB microcontroller
  - 32KB FLASH, 2.5KB SRAM, 1KB EEPROM
  - 12 10-bit ADC channels
  - Serial USART, SPI, and TWI (I2C) communications
  - 4 timers with 14 PWM channels (not all simultaneous)
- Mini USB connector
- DFU bootloader preinstalled (program chip over USB)
- ISP header (program chip using external programmer)
- 16MHz crystal
- Green Status LED
- Reset button
- Bootloader selection jumper
- Powered by USB or external power supply
  - 5V (USB) or 3V - 5.5V (external)
- All pins routed to headers (including those used by on-board hardware)
- Can be mounted on a breadboard
- USB pins routed to header pins (for panel-mount USB connector)
- Analog power filter with separate ground pin
- High-quality PCB with gold-plated finish
- Measures approx. 2.1” x 0.9” (52mm x 23mm) and 0.062” (1.6mm) thick.

ATmega32U4 Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  – 135 Powerful Instructions – Most Single Clock Cycle Execution
  – 32 x 8 General Purpose Working Registers
  – Fully Static Operation
  – Up to 16 MIPS Throughput at 16 MHz
  – On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  – 32K Bytes of In-System Self-Programmable Flash
  – 2.5K Bytes Internal SRAM
  – 1K Bytes Internal EEPROM
  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  – Data retention: 20 years at 85°C/ 100 years at 25°C
  – Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
    All supplied parts are preprogrammed with a default USB bootloader
  – Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  – Boundary-scan Capabilities According to the JTAG Standard
  – Extensive On-chip Debug Support
  – Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- USB 2.0 Full-speed/Low Speed Device Module with Interrupt on Transfer Completion
  – Complies fully with Universal Serial Bus Specification Rev 2.0
  – Supports data transfer rates up to 12 Mbit/s and 1.5 Mbit/s
  – Endpoint 0 for Control Transfers: up to 64-bytes
  – 6 Programmable Endpoints with IN or Out Directions and with Bulk, Interrupt or Isochronous Transfers
  – Configurable Endpoints size up to 256 bytes in double bank mode
  – Fully independent 832 bytes USB DPRAM for endpoint memory allocation
  – Suspend/Resume Interrupts
  – CPU Reset possible on USB Bus Reset detection
  – 48 MHz from PLL for Full-speed Bus Operation
  – USB Bus Connection/Disconnection on Microcontroller Request
  – Crystal-less operation for Low Speed mode
- Peripheral Features
  – On-chip PLL for USB and High Speed Timer: 32 up to 96 MHz operation
  – One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  – Two 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  – One 10-bit High-Speed Timer/Counter with PLL (64 MHz) and Compare Mode
  – Four 8-bit PWM Channels
  – Four PWM Channels with Programmable Resolution from 2 to 16 Bits
  – Six PWM Channels for High Speed Operation, with Programmable Resolution from 2 to 11 Bits
  – Output Compare Modulator
  – 12-channels, 10-bit ADC (features Differential Channels with Programmable Gain)
  – Programmable Serial USART with Hardware Flow Control
  – Master/Slave SPI Serial Interface
  – Byte Oriented 2-wire Serial Interface
  – Programmable Watchdog Timer with Separate On-chip Oscillator
  – On-chip Analog Comparator
  – Interrupt and Wake-up on Pin Change
  – On-chip Temperature Sensor
- Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection
– Internal 8 MHz Calibrated Oscillator
– Internal clock prescaler & On-the-fly Clock Switching (Int RC / Ext Osc)
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

- I/O and Packages
  – All I/O combine CMOS outputs and LVTTL inputs
  – 26 Programmable I/O Lines
  – 44-lead TQFP Package, 10x10mm
  – 44-lead QFN Package, 7x7mm
- Operating Voltages
  – 2.7 - 5.5V
- Operating temperature
  – Industrial (-40°C to +85°C)
- Maximum Frequency
  – 8 MHz at 2.7V - Industrial range
  – 16 MHz at 4.5V - Industrial range



About

Contact Information

Justin Mattair
MattairTech LLC
PO Box 1079
Heppner, OR 97836 USA
541-626-1531
justin@mattair.net
http://www.mattairtech.com/

Support Information

Please check the MattairTech website (http://www.MattairTech.com/) for support information, including firmware and documentation updates. Please feel free to contact me using the above email address or phone number (email is best). Support for AVRs in general can be found at AVRfreaks (http://www.avrfreaks.net/). There, I monitor the forums section as the user physicist.

Precautions

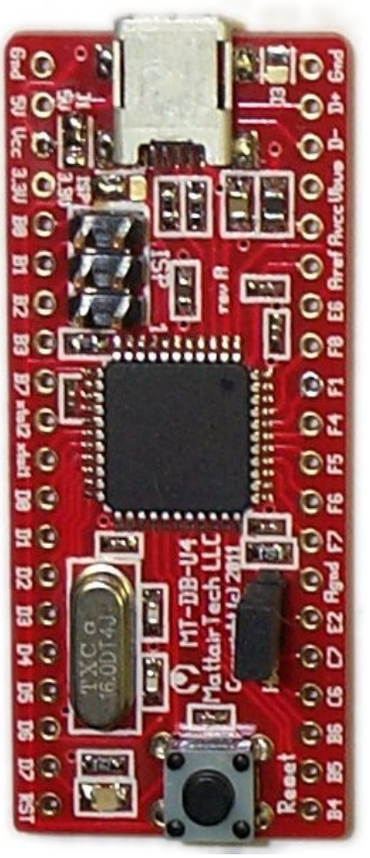
CAUTION
Do not change power configuration while unit is powered. Do not short 5V, Vbus, 3.3V, Avcc, or ground to each other. When connecting Aref externally, connect to a voltage source <= Vcc and be sure that the internal reference is disabled.

CAUTION
The MT-DB-U4 contains static sensitive components. Use the usual ESD procedures when handling.

CAUTION
Improper fuse settings may result in an unusable AVR. Be certain that you know the effects of changing the fuses, that you understand the convention used for describing the state of the fuses (programmed = 0), and that you are using an appropriate programming speed before attempting to change fuse settings.

MT-DB-U4 Hardware

Layout / Header Pins



Pin Descriptions

Pin	Description
Gnd	Digital ground
5V, Vbus	5V output from USB Vbus. Vbus pin is tied to 5V pin. These pins are connected to the Vbus and UVcc pins of the microcontroller.
Vcc	Voltage input pin. Use solder jumper J1 to connect this pin to 5V (default setting) when using USB power. In this case, Vcc is an output. Leave J1 unconnected to supply power from an external source to the Vcc pin. This pin is connected to the Vcc and AVcc pins on the microcontroller, as well as the ISP header and reset pullup. See Power Configuration Section.
3.3V	3.3V output from the microcontrollers internal 3.3V regulator. This pin is connected to Ucap on the microcontroller.
Avcc	Voltage input to the analog section of the microcontroller. This pin is connected to Vcc through a 10uH inductor and 100nF capacitor and provides power to the microcontroller analog section (Avcc pin).
Aref	Voltage input. This is the reference voltage used by the ADC in the microcontroller. **DO NOT** connect if using an internal reference.
Agnd	Analog ground
RST	Connects to reset pin of microcontroller as well as the reset button. A 10K pullup resistor and 100nF capacitor are connected to this pin.
E2 / HWB	This pin is connected to the HWB jumper. The jumper is connected to ground through a 240 ohm resistor. The pin is sampled after all reset sources, including power-up. If the pin is low (HWB jumper installed), then the bootloader is run. If the pin is high (HWB jumper removed), then the user application is run. This pin functions as a normal GPIO pin at all other times. The 240 ohm resistor provides short-circuit protection in case the pin is used as an output and the jumper is installed.
D7 / LED	The green status LED is connected to this pin. The LED is connected to ground through a 240 ohm resistor. The user application is free to use this LED. Drive the pin high to turn on the LED.
xtal1	This pin is connected to the on-board 16MHz crystal. If using an external clock, connect it to this pin and configure the microcontroller fuses to use an external clock. This is also useful for HV programming or recovery from incorrectly set fuses.
xtal2	This pin is connected to the on-board 16MHz crystal. This pin is useful along with xtal1 to connect an external crystal. In this case, you must uninstall the on-board crystal.
E6, F0, F1, F4, F5, F6, F7	These are generally used as analog pins. Analog ground plane runs under these pins. Consult datasheet for functionality.
All other pins	Consult datasheet for functionality.

Integrated Dual-Axis Gyro

IDG-500

FEATURES

- Integrated X- and Y-axis gyros on a single chip
- Two separate outputs per axis for standard and high sensitivity:
  - X-/Y-Out Pins: 500°/s full scale range, 2.0mV/°/s sensitivity
  - X/Y4.5Out Pins: 110°/s full scale range, 9.1mV/°/s sensitivity
- Integrated amplifiers and low-pass filters
- Auto-Zero function
- On-chip temperature sensor
- High vibration rejection over a wide frequency range
- High cross-axis isolation by proprietary MEMS design
- 3V single-supply operation
- Hermetically sealed for temp and humidity resistance
- 10,000 g shock tolerant
- Smallest dual axis gyro package at 4 x 5 x 1.2mm
- RoHS and Green Compliant

APPLICATIONS

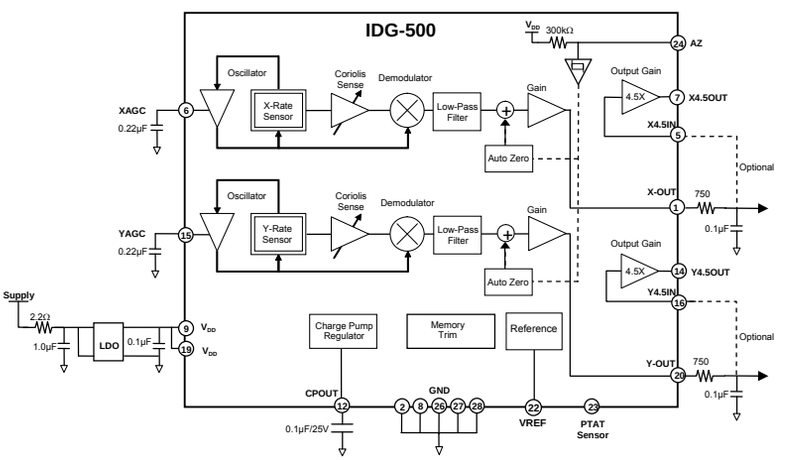
- General Motion Sensing
- Vehicle Motion Analysis
- Platform Stabilization
- Inertial Measurement Units

GENERAL DESCRIPTION

The IDG-500 is an integrated dual-axis angular rate sensor (gyroscope). It uses InvenSense's proprietary and patented MEMS technology with vertically driven, vibrating masses to make a functionally complete, low-cost, dual-axis angular rate sensor. All required electronics are integrated onto a single chip with the sensor.

The IDG-500 gyro uses two sensor elements with novel vibrating dual-mass bulk silicon configurations that sense the rate of rotation about the X- and Y-axis (in-plane sensing). This results in a unique, integrated dual-axis gyro with guaranteed-by-design vibration rejection and high cross-axis isolation. It is specifically designed for demanding consumer applications requiring low cost, small size and high performance.

The IDG-500 gyro includes the integrated electronics necessary for application-ready functionality. It incorporates X- and Y-axis low-pass filters and an EEPROM for on-chip factory calibration of the sensor. Factory trimmed scale factors eliminate the need for external active components and end-user calibration. This product is lead-free and Green Compliant.



SPECIFICATIONS

All parameters specified are @ VDD = 3.0 V and Ta = 25°C. External LPF @ 2kHz. All specifications apply to both axes.

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
SENSITIVITY						
Full-Scale Range	At X-OUT and Y-OUT			±500		°/s
	At X4.5Out and Y4.5Out			±110		°/s
Sensitivity	At X-OUT and Y-OUT			2.0		mV/°/s
	At X4.5Out and Y4.5Out			9.1		mV/°/s
Initial Calibration Tolerance	At X-OUT and Y-OUT			±6		%
Over Specified Temperature	At X-OUT and Y-OUT			±10		%
Nonlinearity	At X-OUT and Y-OUT, Best Fit Straight Line			<1		% of FS
Cross-axis Sensitivity				±1		%
REFERENCE						
Voltage (VREF)				1.35		V
Tolerance				±50		mV
Load Drive				100		µA
Capacitive Load Drive	Load directly connected to VREF			100		pF
Power Supply Rejection	VDD= 2.7V to 3.3V			1		mV/V
Over Specified Temperature				±5		mV
ZERO-RATE OUTPUT						
Static Output (Bias)	Factory Set			1.35		V
Initial Calibration Tolerance	Relative to VREF	With Auto Zero		±20		mV
		Without Auto Zero		±250		
Over Specified Temperature	Relative to VREF	Without Auto Zero		±50		mV
Power Supply Sensitivity	@ 50 Hz			10		°/sec/V
FREQUENCY RESPONSE						
High Frequency Cutoff	Internal LPF -90°			140		Hz
LPF Phase Delay	10Hz			-4.5		°
MECHANICAL FREQUENCIES						
X-Axis Resonant Frequency			20	24	28	kHz
Y-Axis Resonant Frequency			23	27	31	kHz
Frequency Separation	X and Y Gyroscopes			3		kHz
NOISE PERFORMANCE						
Total RMS Noise	Bandwidth 1Hz to 1kHz, At X-OUT and Y-OUT			0.8		mV rms
OUTPUT DRIVE CAPABILITY						
Output Voltage Swing	Load = 100kΩ to $V_{dd}/2$		0.05		Vdd-0.05	V
Capacitive Load Drive				100		pF
Output Impedance				100		Ω
POWER ON-TIME						
Zero-rate Output	Settling to ±3°/s			50	200	ms
AUTO ZERO CONTROL						
Auto Zero Logic High	Rising Input			1.9		V
Auto Zero Logic Low	Falling Input			0.9		V
Auto Zero Pulse Duration			2		1500	µsec
Offset Settle Time After Auto Zero				7		msec

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
POWER SUPPLY (VDD)					
Operating Voltage Range		2.7	3.0	3.3	V
Quiescent Supply Current			7		mA
Over Specified Temperature			±2		mA
TEMPERATURE SENSOR					
Sensitivity	Range -20 to +85°C		4		mV/°C
Offset			1.25		V
Output Impedance			12		kΩ
TEMPERATURE RANGE					
Specified Temperature Range		-20		+85	°C

RECOMMENDED OPERATING CONDITIONS

Parameter	Min	Typical	Max	Unit
Power Supply Voltage (VDD)	2.7	3.0	3.3	V
Power Supply Voltage (VDD) Rise Time (10% - 90%)			20	ms

ABSOLUTE MAXIMUM RATINGS

Stress above those listed as “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameter	Rating
Power Supply Voltage (VDD)	-0.3V to +6.0V
Acceleration (Any Axis, unpowered)	10,000*g* for 0.3ms
Operating Temperature Range	-40 to +105°C
Storage Temperature Range	-40 to +125°C

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

3-axis sensing
Small, low profile package
4 mm × 4 mm × 1.45 mm LFCSP
Low power : 350 µA (typical)
Single-supply operation: 1.8 V to 3.6 V
10,000 *g* shock survival
Excellent temperature stability
BW adjustment with a single capacitor per axis
RoHS/WEEE lead-free compliant

APPLICATIONS

Cost sensitive, low power, motion- and tilt-sensing applications
Mobile devices
Gaming systems
Disk drive protection
Image stabilization
Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

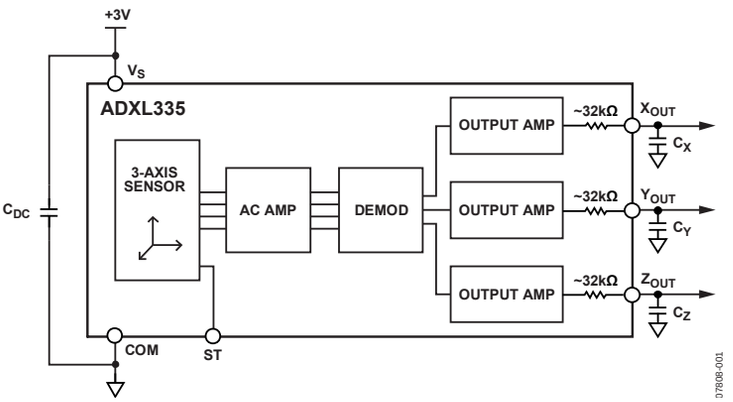


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

SPECIFICATIONS

T_A = 25°C, V_S = 3 V, C_X = C_Y = C_Z = 0.1 µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		m*g*/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µ*g*/√Hz rms
Noise Density Z_{OUT}			300		µ*g*/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × C). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.
[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately 160 × C_X or C_Y or C_Z + 1 ms, where C_X, C_Y, C_Z are in microfarads (µF).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered)	10,000 *g*
Acceleration (Any Axis, Powered)	10,000 *g*
V_S	−0.3 V to +3.6 V
All Other Pins	(COM − 0.3 V) to (V_S + 0.3 V)
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Temperature Range (Powered)	−55°C to +125°C
Temperature Range (Storage)	−65°C to +150°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 µF capacitor, C_{DC}, placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply. However, in applications where noise is present at the 50 kHz internal clock frequency (or any harmonic thereof), additional care in power supply bypassing is required because this noise can cause errors in acceleration measurement.

If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (1 µF or greater) can be added in parallel to C_{DC}. Ensure that the connection from the ADXL335 ground to the power supply ground is low impedance because noise transmitted through ground has a similar effect to noise transmitted through V_S.

SETTING THE BANDWIDTH USING C_X, C_Y, AND C_Z

The ADXL335 has provisions for band limiting the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is

$$F_{-3\,dB} = 1/(2\pi(32\ k\Omega) \times C_{(X,\,Y,\,Z)})$$

or more simply

$$F_{-3\,dB} = 5\ \mu F/C_{(X,\,Y,\,Z)}$$

The tolerance of the internal resistor (R_{FILT}) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 0.0047 µF for C_X, C_Y, and C_Z is recommended in all cases.

Table 4. Filter Capacitor Selection, C_X, C_Y, and C_Z

Bandwidth (Hz)	Capacitor (µF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

SELF-TEST

The ST pin controls the self-test feature. When this pin is set to V_S, an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is −1.08 *g* (corresponding to −325 mV) in the X-axis, +1.08 *g* (or +325 mV) on the Y-axis, and +1.83 *g* (or +550 mV) on the Z-axis. This ST pin can be left open-circuit or connected to common (COM) in normal use.

Never expose the ST pin to voltages greater than V_S + 0.3 V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low V_F clamping diode between ST and V_S is recommended.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The selected accelerometer bandwidth ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor to improve the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at X_{OUT}, Y_{OUT}, and Z_{OUT}.

The output of the ADXL335 has a typical bandwidth of greater than 500 Hz. The user must filter the signal at this point to limit aliasing errors. The analog bandwidth must be no more than half the analog-to-digital sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL335 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of µg/√Hz (the noise is proportional to the square root of the accelerometer bandwidth). The user should limit bandwidth to the lowest frequency needed by the application to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL335 is determined by

$$rms\ Noise = Noise\ Density \times (\sqrt{BW \times 1.6})$$

It is often useful to know the peak value of the noise. Peak-to-peak noise can only be estimated by statistical methods. Table 5 is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table 5. Estimation of Peak-to-Peak Noise

Peak-to-Peak Value	% of Time That Noise Exceeds Nominal Peak-to-Peak Value
2 × rms	32
4 × rms	4.6
6 × rms	0.27
8 × rms	0.006

USE WITH OPERATING VOLTAGES OTHER THAN 3 V

The ADXL335 is tested and specified at V_S = 3 V; however, it can be powered with V_S as low as 1.8 V or as high as 3.6 V. Note that some performance parameters change as the supply voltage is varied.

The ADXL335 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At V_S = 3.6 V, the output sensitivity is typically 360 mV/*g*. At V_S = 2 V, the output sensitivity is typically 195 mV/*g*.

The zero *g* bias output is also ratiometric, thus the zero *g* output is nominally equal to $V_S/2$ at all supply voltages.

The output noise is not ratiometric but is absolute in volts; therefore, the noise density decreases as the supply voltage increases. This is because the scale factor (mV/*g*) increases while the noise voltage remains constant. At V_S = 3.6 V, the X-axis and Y-axis noise density is typically 120 μ*g*/√Hz, whereas at V_S = 2 V, the X-axis and Y-axis noise density is typically 270 μ*g*/√Hz.

Self-test response in *g* is roughly proportional to the square of the supply voltage. However, when ratiometricity of sensitivity is factored in with supply voltage, the self-test response in volts is roughly proportional to the cube of the supply voltage. For example, at V_S = 3.6 V, the self-test response for the ADXL335 is approximately −560 mV for the X-axis, +560 mV for the Y-axis, and +950 mV for the Z-axis.

At V_S = 2 V, the self-test response is approximately −96 mV for the X-axis, +96 mV for the Y-axis, and −163 mV for the Z-axis.

The supply current decreases as the supply voltage decreases. Typical current consumption at V_S = 3.6 V is 375 μA, and typical current consumption at V_S = 2 V is 200 μA.

AXES OF ACCELERATION SENSITIVITY

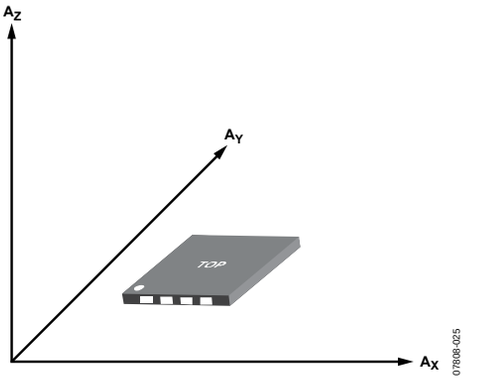


Figure 23. Axes of Acceleration Sensitivity; Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis.

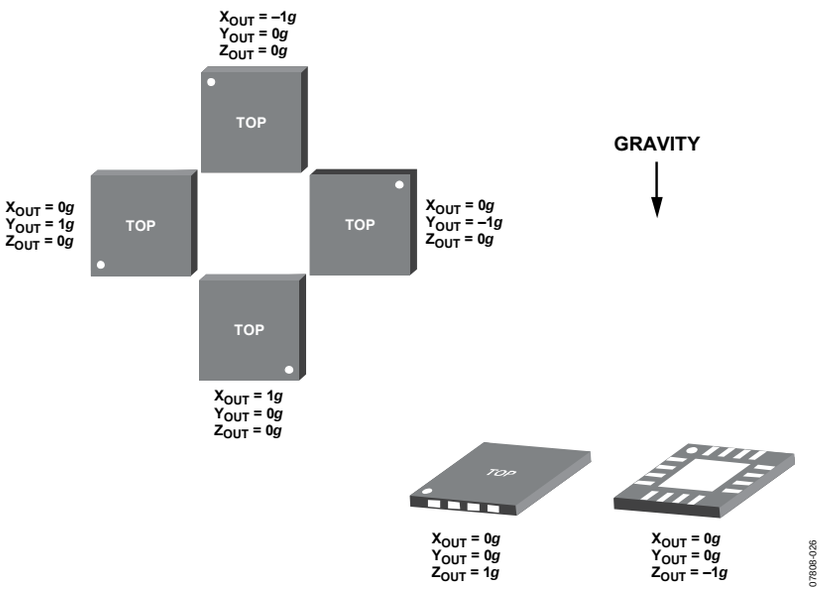


Figure 24. Output Response vs. Orientation to Gravity

LAYOUT AND DESIGN RECOMMENDATIONS

The recommended soldering profile is shown in Figure 25 followed by a description of the profile features in Table 6. The recommended PCB layout or solder land drawing is shown in Figure 26.

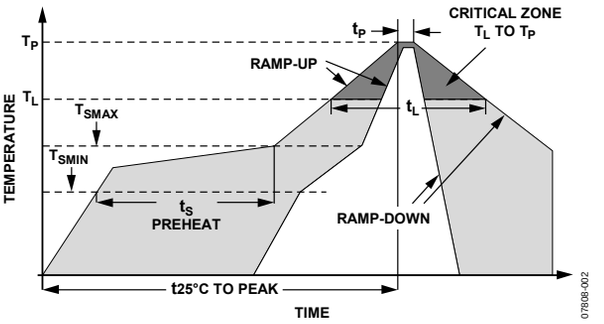


Figure 25. Recommended Soldering Profile

Table 6. Recommended Soldering Profile

Profile Feature	Sn63/Pb37	Pb-Free
Average Ramp Rate (T_L to T_P)	3°C/sec max	3°C/sec max
Preheat		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
Time (T_{SMIN} to T_{SMAX})(t_S)	60 sec to 120 sec	60 sec to 180 sec
T_{SMAX} to T_L		
Ramp-Up Rate	3°C/sec max	3°C/sec max
Time Maintained Above Liquidous (T_L)		
Liquidous Temperature (T_L)	183°C	217°C
Time (t_L)	60 sec to 150 sec	60 sec to 150 sec
Peak Temperature (T_P)	240°C + 0°C/−5°C	260°C + 0°C/−5°C
Time Within 5°C of Actual Peak Temperature (t_P)	10 sec to 30 sec	20 sec to 40 sec
Ramp-Down Rate	6°C/sec max	6°C/sec max
Time 25°C to Peak Temperature	6 minutes max	8 minutes max

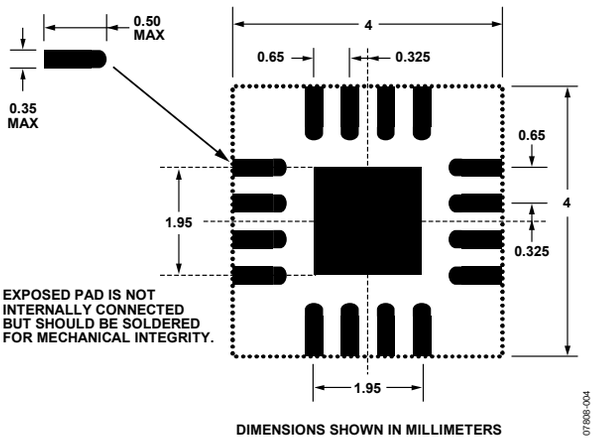
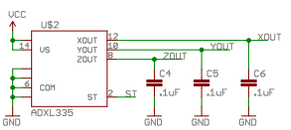
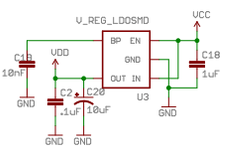


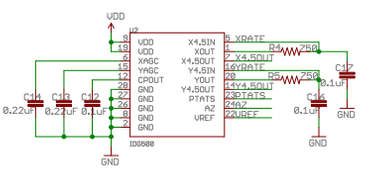
Figure 26. Recommended PCB Layout

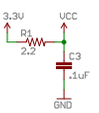
Accelerometer

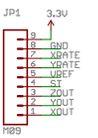


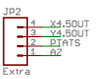
Gyro











SFE

TITLE: 5DOF-v25

Document Number:

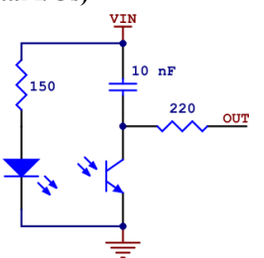
REV:

Date: 9/8/2009 1:57:44 PM

Sheet: 1/1

1. QTR-xRC Sensor Output (Intended for Digital I/Os)

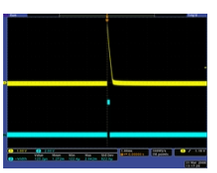
Each **QTR-1RC** [http://www.pololu.com/catalog/product/959] and **QTR-8RC** [http://www.pololu.com/catalog/product/961] reflectance sensor phototransistor output is tied to a capacitor discharge circuit as shown on the right, which allows a digital I/O line on a microcontroller to take an analog reflectance reading by measuring the discharge time of the capacitor. When you have a microcontroller's digital I/O connected to a sensor output, the typical sequence for reading that sensor is:



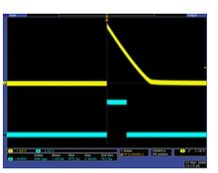
QTR-1RC reflectance sensor schematic diagram.

1. Set the I/O line to an output and drive it high
2. Allow at least 10 us for the 10 nF capacitor to charge
3. Make the I/O line an input (high impedance)
4. Measure the time for the capacitor to discharge by waiting for the I/O line to go low

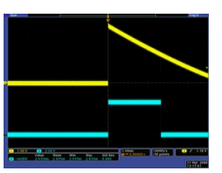
The following three oscilloscope screen captures below demonstrate the result of this procedure. The sensor was positioned 1/8" above a whiteboard-like surface with a 3/4" thick piece of black electrical tape on it. The first reading was taken over the white portion of the surface, the second reading was taken at the edge of the tape, and the third was taken while fully over the black tape. The yellow oscilloscope channel is the sensor output and the blue oscilloscope channel is the output of a mega168 AVR microcontroller representing its interpretation of the sensor output. A 5 V blue signal indicates that the AVR is measuring the sensor output as "high"; a 0 V blue signal indicates that the AVR is measuring the sensor output as "low". In an actual application, the important value is the width of the positive blue pulse. As you can see from the screen captures, the shorter the pulse, the more reflective the surface. Medium-width pulses occur from moderately reflective surfaces, or as you transition from a white surface to a black surface (or vice versa).



QTR-1RC output (yellow) when 1/8" above a white surface and microcontroller timing of that output (blue).



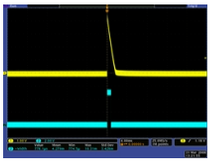
QTR-1RC output (yellow) when 1/8" above a white/black interface and mcu timing of that output (blue).



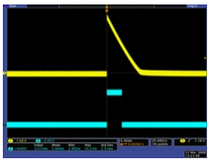
QTR-1RC output (yellow) when 1/8" above a black line and microcontroller timing of that output (blue).

Please note that these data are affected by the specifics of the test. Reflectances depend on the surfaces, and each microcontroller will have its own trip-low threshold. In our specific example, you can see that at a height of 1/8" above our surface, white results in a high pulse width of 120 us and black results in a high pulse width of 2.6 ms. As the pulse width varies between 120 us and 2.6 ms, you can tell that you are approaching or leaving the line.

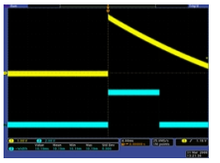
The screen captures below show the results of the same test conducted at a sensor height of 3/8".



QTR-1RC output (yellow) when 3/8" above a white surface and microcontroller timing of that output (blue).



QTR-1RC output (yellow) when 3/8" above a white/black interface and mcu timing of that output (blue).

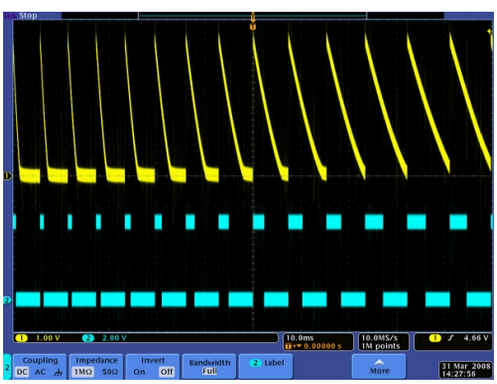


QTR-1RC output (yellow) when 3/8" above a black line and microcontroller timing of that output (blue).

Raising the sensor decreases the overall reflectance of the surface, which in turn lengthens all of the positive pulse widths. This means we need more time to measure the sensor outputs and hence we are limited to lower update rates. At this height, our white surface results in a high pulse width of 780 us and our black surface results in a high pulse width of 10 ms.

Note that at the start of each yellow pulse, there is a 10 us period where our microcontroller is driving the sensor output line high.

Lastly, the following screen capture shows an example of what the sensor output might look like as it sweeps across a black line on a white surface. A motor was used to rotate a white paper disk with a piece of black electrical tape on it in front of the sensor. The electrical noise present in the screen capture is from this motor. If you will be using these sensors in electrically noisy environments (e.g. around motors), you should filter the signal either with a low-pass filter circuit or in your microcontroller software. For example, when timing the high (blue) pulse, wait until the signal stays low for a minimum duration (e.g. 10 us) before accepting the low signal as the end of the pulse.



Example series of QTR-1RC output signals generated as a black line on a spinning white disk passes in front of the sensor.

GP2Y0D810Z0F

Distance Measuring Sensor Unit
Digital output (100 mm) type

■Description

GP2Y0D810Z0F is distance measuring sensor unit, composed of an integrated combination of PD (photo diode) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
The output voltage of this sensor stays high in case an object exists in the specified distance range. So this sensor can also be used as proximity sensor.

■Features

1. Digital output type
2. Short distance type
   Detecting distance : Typ. 100 mm
3. Low profile
   Package size : 13.6×7×7.95 mm
4. Consumption current : Typ. 5 mA
5. Battery drive possible
   Supply voltage : 2.7 to 6.2 V
6. Sunlight tolerance
7. Add Vin terminal, and an external transistor of Vcc line is unnecessary at intermittent operating.

■Agency approvals/Compliance

1. Compliant with RoHS directive (2002/95/EC)

■Applications

1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Robot cleaner

Sheet No.: E4-A01101EN
Date Dec.01.2006

■Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	-0.3 to +7	V
Output terminal voltage	V_O	-0.3 to V_{CC}+0.3	V
Input terminal voltage	V_{in}	-0.3 to V_{CC}+0.3	V
Operating temperature	T_{opr}	-10 to +60	℃
Storage temperature	T_{stg}	-20 to +70	℃
* Soldering temperature	T_{sol}	260	℃

* 5s or less/time up 2times
t = 1.0 mm One side board mounting

■Electro-optical Characteristics

(T_a=25℃,V_{CC}=5V)

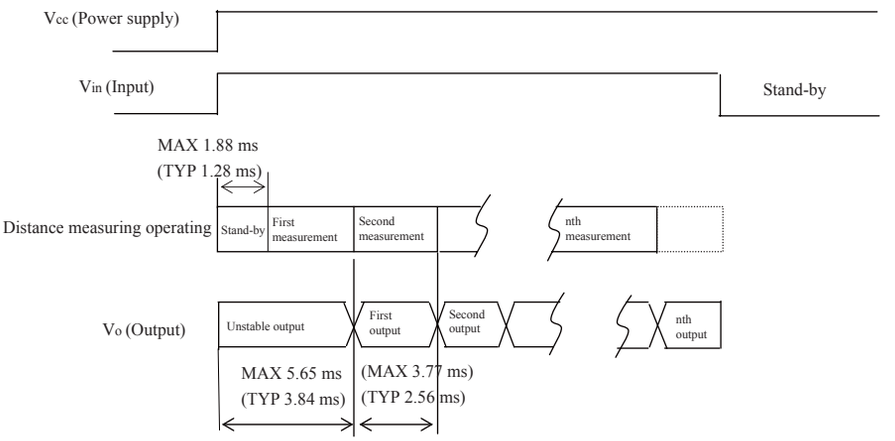
Parameter	Symbol	Rating	MIN.	TYP.	MAX.	Unit
Average supply current	I_{CC1}	V_{CC}=5V, V_{in}=5V, R_1=4.3Ω (*1)	—	5	6.5	mA
Average supply current	I_{CC2}	V_{CC}=5V, V_{in}=5V, R_1=4.3Ω (*1)	—	9	10.5	mA
Stand-by supply current	I_{CC3}	V_{CC}=5V, V_{in}=0V	—	5	8	μA
Output voltage	V_{OH}	Output voltage at high level	V_{CC}-0.6	—	—	V
	V_{OL}	Output voltage at low level	—	—	0.6	V
Detecting distance	L	(*2)(*3)	80	100	130	mm

(*1) I_{CC1} : (LED Emitting time : Typ. 20 μs × 8 times), I_{CC2} : (Emitting time : Typ. 20 μs × 15 times),
LED Pulse Current : Typ. 70 mA
(*2) Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27·white face, reflectance ; 90%)
(*3) Output voltage switch has a hysteresis width. The distance specified by L should be
the distance which the output turns from L to H in case an object moves to the sensor.

■Recommended operating conditions

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{CC}		2.7 to 6.2	V
High level input voltage	V_{inH}	CMOS level signal. Operating	MIN V_{CC}-0.2	V
Low level input voltage	V_{inL}	CMOS level signal. Standby state	MAX 0.2	V

Fig. 1 Timing chart



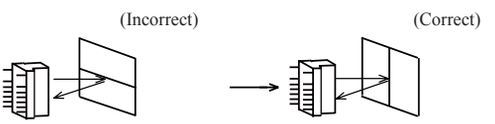
■Notes

●Advice for the optics

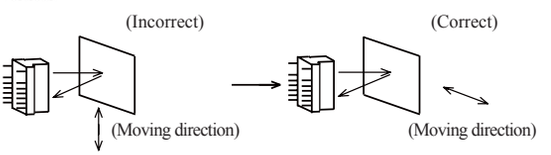
- The lens of this device needs to be kept clean. There are cases that dust, water or oil and so on deteriorate the characteristics of this device. Please consider in actual application.
- Please don't do washing. Washing may deteriorate the characteristics of optical system and so on. Please confirm resistance to chemicals under the actual usage since this product has not been designed against washing.

●Advice for the characteristics

- In case that an optical filter is set in front of the emitter and detector portion, the optical filter which has the most efficient transmittance at the emitting wavelength range of LED for this product ($\lambda = 870 \pm 70$nm), shall be recommended to use. Both faces of the filter should be mirror polishing. Also, as there are cases that the characteristics may not be satisfied according to the distance between the protection cover and this product or the thickness of the protection cover, please use this product after confirming the operation sufficiently in actual application.
- In case that there is an object near to emitter side of the sensor between sensor and a detecting object, please use this device after confirming sufficiently that the characteristics of this sensor do not change by the object.
- When the detector is exposed to the direct light from the sun, tungsten lamp and so on, there are cases that it can not measure the distance exactly. Please consider the design that the detector is not exposed to the direct light from such light source.
- Distance to a mirror reflector can not be sometimes measured exactly. In case of changing the mounting angle of this product, it may measure the distance exactly.
- In case that reflective object has boundary line which material or color etc. are excessively different, in order to decrease deviation of measuring distance, it shall be recommended to set the sensor that the direction of boundary line and the line between emitter center and detector center are in parallel.

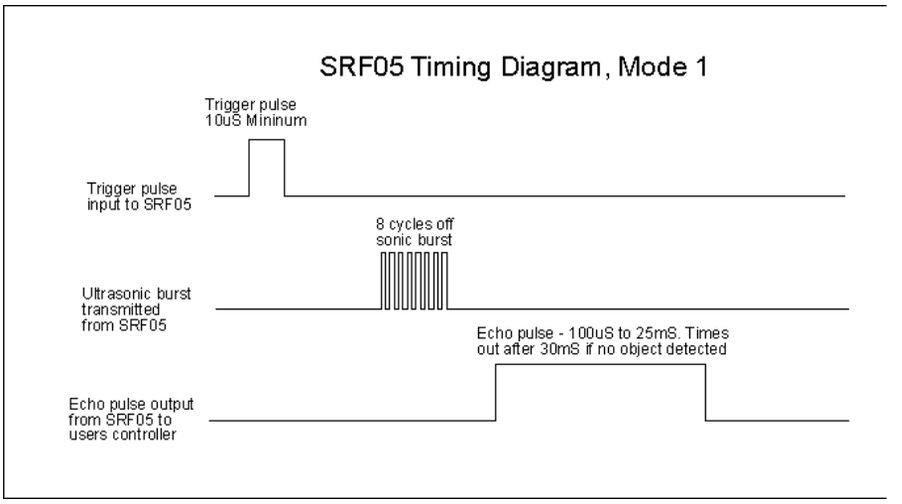


- In order to decrease deviation of measuring distance by moving direction of the reflective object, it shall be recommended to set the sensor that the moving direction of the object and the line between emitter center and detector center are vertical.



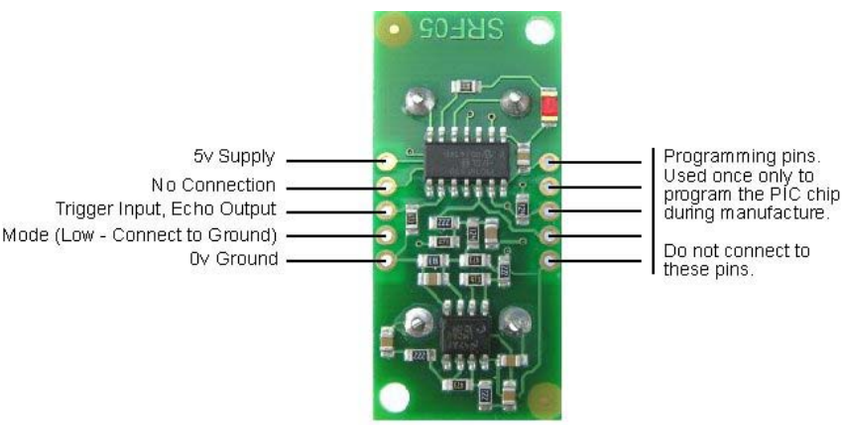
●Notes on handling

- There are some possibilities that the internal components in the sensor may be exposed to the excessive mechanical stress. Please be careful not to cause any excessive pressure on the sensor package and also on the PCB while assembling this product.
- Soldering shall be done with a soldering iron and below 260℃, less than 5s and maximum 2 times. Also, please pay attention not to put outer force on lead terminals while soldering. Please do not apply flow soldering because it may damage optical lens of the device.

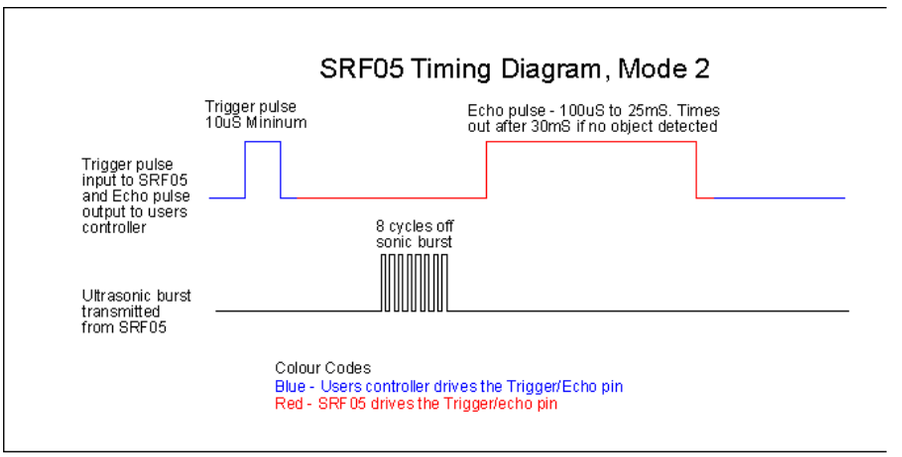


Mode 2 - Single pin for both Trigger and Echo

This mode uses a single pin for both Trigger and Echo signals, and is designed to save valuable pins on embedded controllers. To use this mode, connect the mode pin to the 0v Ground pin. The echo signal will appear on the same pin as the trigger signal. The SRF05 will not raise the echo line until 700uS after the end of the trigger signal. You have that long to turn the trigger pin around and make it an input and to have your pulse measuring code ready. The PULSIN command found on many popular controllers does this automatically.



Connections for single pin Trigger/Echo Mode



To use mode 2 with the Basic Stamp BS2, you simply use PULSOUT and PULSIN on the same pin, like this:

```
SRF05 PIN 15            ' use any pin for both trigger and echo
Range VAR Word          ' define the 16 bit range variable

SRF05 = 0               ' start with pin low
PULSOUT SRF05, 5        ' issue 10uS trigger pulse (5 x 2uS)
PULSIN SRF05, 1, Range  ' measure echo time
Range = Range/29        ' convert to cm (divide by 74 for inches)
```

Calculating the Distance
The SRF05 Timing diagrams are shown above for each mode. You only need to supply a short 10uS pulse to the trigger input to start the ranging. The SRF05 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high (or trigger line in mode 2). It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else. If nothing is detected then the SRF05 will lower its echo line anyway after about 30mS.

The SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. uS/58=cm or uS/148=inches.

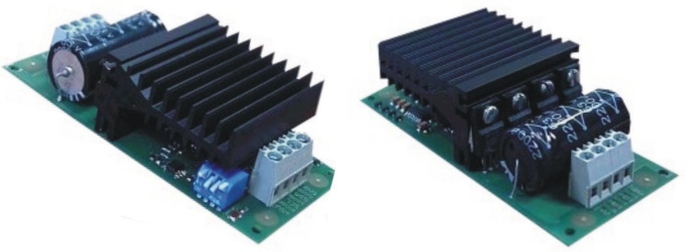
The SRF05 can be triggered as fast as every 50mS, or 20 times each second. You should wait 50ms before the next trigger, even if the SRF05 detects a close object and the echo pulse is shorter. This is to ensure the ultrasonic "beep" has faded away and will not cause a false echo on the next ranging.

Alda. Mazarredo Nº 47 –1º Dpto 2
48009 Bilbao – Vizcaya (Spain)
Tfno/Fax: 94 4230651
www.microcontroladores.cozm
info@microcontroladores.com

MSE Microsystems Engineering

1.- DESCRIPCION

El módulo MD03 consiste en un driver para el control de motores DC diseñado por la firma Devantech.Ltd. Basado en un puente H, es capaz de controlar motores de hasta 50V con una intensidad de 20 A. Se muestra en la figura 1.



Entre las características mas relevantes cabe destacar las siguientes:

- Alimentación estándar de +5V para la lógica interna, con un consumo de 50mA.
- Tensión +V de alimentación para el motor. Esta puede variar ser de 5 a 50VDC.
- Diferentes tipos de control sobre el motor, que se seleccionan mediante dip-switch:

  a) Modo analógico de 0V – 2.5V – 5V
  b) Modo analógico de 0V a 5V
  c) Modo RC
  d) Modo I2C

1.1 Modo analógico de 0V – 2.5V – 5V

En este modo el motor se controla con una tensión analógica que puede variar de 0V a 5V y que se aplica por la entrada SDA. La señal SCL no se emplea y debe conectarse a +5V o a GND.

Si la tensión aplicada es de 0V el motor gira en sentido antihorario a la máxima velocidad. Con una tensión de +2.5V el motor queda detenido. Aplicando una tensión de +5V el motor gira a la máxima velocidad en sentido horario.

Una variación de +/- 2.7% alrededor de los 2.5V proporcionan una banda estable en la que el motor permanece detenido.

1.2 Modo analógico de 0V – 5V

El motor se controla mediante la tensión analógica de entrada que se aplica por la entrada SDA y que varía entre 0V (motor parado) y 5V (máxima potencia).

La entrada SCL permite controlar el sentido de giro. Aplicando un nivel lógico “0” se produce un giro antihorario, con nivel “1” el giro es en sentido horario.

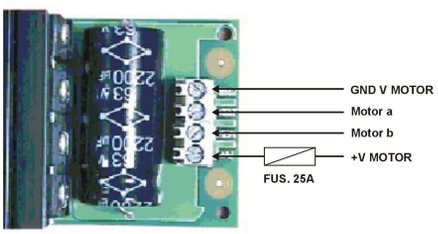
Alda. Mazarredo Nº 47 –1º Dpto 2
48009 Bilbao – Vizcaya (Spain)
Tfno/Fax: 94 4230651
www.microcontroladores.cozm
info@microcontroladores.com

2.- CARACTERISTICAS TECNICAS

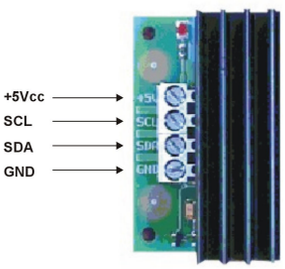
PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	113x52x30	mm
Consumo máximo de la lógica de control	50	mA
Intensidad máxima de salida para el motor	20	A
Preselección del limitador de corriente	20	A
Tensión de alimentación para la lógica del motor	5	V
Tensión de alimentación para el motor	5 – 50	V

3. CONEXIONADO

Se realiza mediante una serie bornas que permiten una fácil conexión. Cada una de las señales vienen indicadas en la propia placa impresa. En la figura 3 se muestran las conexiones relativas al motor. Por la borna superior se conecta la señal GND de alimentación. En las dos bornas centrales, Motor a y Motor b, se conecta el propio motor a controlar. Es posible intercambiar estas dos conexiones para conseguir el sentido de giro horario o antihorario apropiado. Finalmente, mediante la borna inferior, se conecta la tensión de alimentación del motor, +V MOTOR. El valor de esta puede variar de 5V a 50V en función de las necesidades del motor. Es importante conectar un fusible de 25/30 A tal y como se muestra en la figura. Una sobre intensidad puede dañar permanentemente el módulo MD03.



En la figura 4 se muestra la conexión de las señales que controlan el módulo MD03. Se realiza también mediante 4 bornas.



Por la borna superior se aplica la tensión de +5Vcc que alimenta a toda la lógica de control del módulo MD3. Las bornas centrales se emplean para conectar las señales de control SCL y SDA. Según el modo de

trabajo elegido por las mismas se aplican las señales analógicas, la de RC o las señales propias del bus I2C para la transferencia de datos entre el módulo y el Mater. La borna inferior se emplea para conectar la señal GND de alimentación.

4.- AJUSTES

El módulo MD03 no necesita de ningún tipo de ajuste excepto la selección, mediante los dip-switch, del modo de trabajo deseado. Dicho modo se debe establecer antes de conectar la tensión de alimentación. Posteriores variaciones en ese modo de trabajo no tendrán efecto hasta que el módulo vuelva a conectarse.

5.- APLICACIONES

El módulo MD3 está especialmente orientado al control de motores DC de pequeña y mediana potencia. Estos motores puede se empleados en aplicaciones de industriales, robótica y en cualquier sistema que necesite algún tipo de movimiento.

EZURiO

Embedded Intelligent *Bluetooth*™ Serial Module

Part Number: BISMS02BI-01

1. General Description

Ezurio's Embedded Intelligent *Bluetooth* Serial Module is a fully integrated and qualified Class 1 *Bluetooth* solution designed for lowest cost of integration and ownership for designers wishing to include *Bluetooth* functionality in their products. The module is qualified to *Bluetooth* Version 2.0.

The Embedded Intelligent *Bluetooth* Serial Module is designed to give a rugged solution that is ideal for industrial automation and ruggedised handheld devices. It works over a wide temperature range of -40°C to +85°C. The physical form of the module allows designers to mount the antenna section of the module outside a screened enclosure.

The Embedded Intelligent Serial Module is based on Cambridge Silicon Radio's BlueCore4 chipset. The module contains all of the hardware and firmware for a complete *Bluetooth* solution, requiring no further components. The Module has an integrated, high performance antenna which is matched with the *Bluetooth* RF and baseband circuitry. The firmware integrated into the module implements the higher layer *Bluetooth* protocol stack, up to and including the Generic Access Profile (GAP), Service Discovery Profile (SDAP), Serial Port Profile (SPP) and Audio Gateway. A virtual processor is used within the BC04 to implement an AT command processor. This interfaces to the host system over a straight forward serial port using an extensive range of AT commands. The AT command set abstracts the *Bluetooth* protocol from the host application, saving many months of programming and integration time. It provides extremely short integration times for data oriented cable replacement and voice applications. A low cost development system and integrated RS232 products with the same firmware are available for fast product evaluation and development.

An alternative version of firmware is available that provides support for multi-point applications.

The Module can be configured so that it can be attached to a 'dumb' terminal or attached to a PC or PDA for cable replacement applications.

In addition to the *Bluetooth* functionality, The Embedded Intelligent Serial Module provides access to 6 General I/O lines and one ADC input. These can be configured to extend the UART control or to provide connection to simple devices such as switches or LEDs without requiring any external processing. The GPIO lines can be accessed either via the wired host UART connection, or remotely over the *Bluetooth* link. Support is also provided for a PCM connection to an external audio codec.

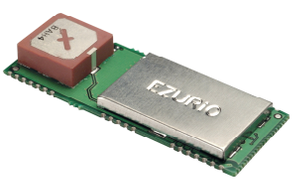
The Embedded Intelligent *Bluetooth* Module is supplied in a small form factor PCB (17.7mm x 46.0mm x 5.0mm), that solders directly. The module includes a high sensitivity, high gain antenna which provides excellent range. Typical open field performance provides ranges of over 250 metres at transmit powers of 4mW.

Support is provided for low power modes that make the Embedded Intelligent *Bluetooth* Module particularly applicable to battery powered installations.

The Embedded Intelligent *Bluetooth* Module is Lead-free and RoHS compliant and supports an industrial operating temperature range of -40°C to +85°C.

1.1 Applications

- POS Equipment
- Industrial Automation
- Vending Equipment
- Automotive Applications
- Telematics
- Medical



Bluetooth is a trademark owned by *Bluetooth* SIG, Inc., USA, and is licensed to Ezurio Ltd

2. Features

Feature	Implementation
Bluetooth Transmission	Class 1
Fully *Bluetooth* pre-qualified	*Bluetooth* 2.0
Range	250 metres typical (free space)
Frequency	2.400 – 2.485 GHz
Max Transmit Power	+6dBm
Min Transmit Power	-27dBm
Receive Sensitivity	Better than -86dB
Data Transfer rate	Up to 300Kbps over UART.
Serial Interface	RS-232 bi-directional for commands and data using AT commands
Serial parameters	Default 9600,n,8,1 - Configurable from 1,200bps to 961,200 bps.
	7 bit firmware is available – please contact Ezurio
	Support for DTR, DSR, DCD, RI, RTS, CTS
Physical size	17.7mm x 46.0mm x 5.0mm, 8g
Current consumption	Less than 36mA during data transfer in standard power mode. Lower powers are attainable with a configurable low power mode.
Low power sniff mode	2.5mA typ
Temperature Range	Normal operation: -40°C to +85°C
Supply Voltage	3.3V – 7.0V
Interface Levels	3.0V Logic
Audio	Audio can be transferred over SCO channels through the PCM interface at 64kbps. PCM can be configured as master or slave.
	Support for dual slave PCM connections.
Profiles	Server - SPP (Full), DUN, Audio Gateway, Headset, Handsfree
	Client - All RFCOMM based profiles
Multipoint	Max 7 slaves with multipoint
Field upgradeable	Over UART
Protocols	Single point firmware is controlled and configured using AT Commands. Standard multipoint firmware uses a simple packet based protocol and requires a host to enable the module to function effectively.
	Single point only allows a point to point connection whereas multipoint allows more than one simultaneous connection.
GPIO	6 x digital (DTR can also be allocated as GPIO)
ADC	1 x 8 bit
Indicators	Pads for 2 programmable LEDs
Lead free	Lead-free and RoHS compliant

3.2 Pin Descriptions

The table below defines the pin functions. Refer to the previous section for the pin location

Pin No.	Signal	Description
1	VCC	
3	USB /RESERVED	Do not connect
5	RESET-	Reset I/P *
7	GPIO_7	I/O for Host
20	GND	
22	UART_RI	I/O for Host
24	UART_RX	Receive Data I/P
26	UART_TX	Transmit Data O/P
28	UART_DSR	Input
30	PCM_SYNC	PCM Sync I/P
32	PCM_CLK	PCM Clock I/P
34	GPIO_9 PCM_SLVCLK	I/O for Host (Slave PCM Clock)
36	GPIO_4	I/O for Host (LED1)
38	GPIO_8	I/O for Host

Pin No.	Signal	Description
2	USB /RESERVED	Do not connect
4	GND	
6	GPIO_6	I/O for Host
8-19	N/C	Do not Connect
21	UART_DCD	I/O for Host
23	UART_RTS	Request to Send O/P
25	UART_CTS	Clear to Send I/P
27	UART_DTR (GPIO_3)	I/O for Host
29	GND	
31	PCM_IN	PCM Data I/P
33	PCM_OUT	PCM Data O/P
35	GPIO_5	I/O for Host (LED2)
37	Analogue_0	ADC
39-42	N/C	Do not connect

Notes:

* The reset line has a fixed 10kOhm pull up resistor with the reset being active low.

PIO lines can be configured through software to be either inputs or outputs with weak or strong pull-ups or pull-downs. At reset, all PIO lines are configured as inputs with weak pull-downs.

UART_RX, UART_TX, UART_CTS, UART_RTS, UART_RI, UART_DCD and UART_DSR are all 3.0v level logic. For example, when RX and TX are idle they will be sitting at 3.0V. Conversely for handshaking pins CTS, RTS, RI, DCD, DSR a 0v is treated as an assertion.

Pin 22 (UART_RI) is active low. It is normally 3.0v. When a remote device initiates a connection, this pin goes low. This means that when this pin is converted to RS232 voltage levels it will have the correct voltage level for assertion.

Pin 21 (UART_DCD) is active low. It is normally 3.0v. When a connection is live this pin is low. This means that when this pin is converted to RS232 voltage levels it will have the correct voltage level for assertion.

Pin 28 (UART_DSR) is an input, with active low logic. It should be connected to the DTR output of the host. When the Module is in high speed mode (See definition for S Register 507), this pin should be asserted by the host to ensure that the connection is maintained. A deassertion is taken to mean that the connection should be dropped, or an online command mode is being requested.

The GPIO pins can be accessed using S Registers in the range 623 to 629. GPIO4 and 5 are connected to unpopulated LED pads on the module. If these I/O pins are set for input, then the LED will be driven by the host and appropriate drive current requirements must be satisfied. A Logic 1 switches on the LED.

GPIO3 is shares the pin with DTR output (active low). See S Register 552 & 553.

Analogue 0 input should not exceed 1.8v. S Register 701 is used to access it.

3.3 Electrical Specifications

3.3.1 Absolute Maximum ratings

Absolute maximum ratings for supply voltage and voltages on digital and analogue pins of the Module are listed below; exceeding these values will cause permanent damage.

Parameter	Min	Max	Unit
Peak current of power supply	0	100	mA
Voltage at digital pins	-0.3	3.3	V
Voltage at POWER pin	3.3	7	V

3.3.2 Recommended Operating Parameters

3.3.2.1 Power Supply

Signal Name	Pin No	I/O	Voltage level	Comments
Vcc	1	I	3.3V to 7.0V	I_{typ} = 30mA
GND	4, 20, 29			

3.3.2.2 RS-232 Interface

Signal Name	Pin No	I/O	Signal level	Comments
UART_TX	26	O	V_{OL}max=0.2V V_{OH}min=2.8V	
UART_RX	24	I	V_{IL}max=0.8V V_{IH}min=2.1V V_{IH}max=3.4V	
UART_CTS	25	I	V_{IL}max=0.8V V_{IH}min=2.1V V_{IH}max=3.4V	
UART_RTS	23	O	V_{OL}max=0.2V V_{OH}min=2.8V	
UART_DSR	28	I	V_{IL}max=0.8V V_{IH}min=2.1V V_{IH}max=3.4V	
UART_DTR	27	O	V_{OL}max=0.2V V_{OH}min=2.8V	Shared with GPIO_3
UART_RI	22	I or O	O/P : V_{OL}max=0.2V V_{OH}min=2.8V I/P : V_{IL}max=0.8V V_{IH}min=2.1V V_{IH}max=3.4V	Direction may be programmed.
UART_DCD	21	I or O	O/P : V_{OL}max=0.2V V_{OH}min=2.8V I/P : V_{IL}max=0.8V V_{IH}min=2.1V V_{IH}max=3.4V	Direction may be programmed.

5. DC Characteristics

5.1 RF Performance

5.1.1 Transmit Power

Conducted Transmit Power	min: 1.0mW (0dBm)	max: 4mW (6dBm)
Effective Transmit Power	min:0dBm	Max: +6dBm

Output power can be reduced by program control

5.1.2 Receive Sensitivity

Receive Sensitivity	-86dBm (at 25°C)
Antenna Gain	+2dBi typ
Effective Receive Sensitivity	-88dBm (at 25°C)

5.2 Range

Range is determined by the environment and the orientation of the module.

The data throughput of the Module is limited to 300Kbps by the parsing of the data being transferred through the RFCOMM stack.

6. Functional Description

The Embedded Intelligent *Bluetooth* module is a self-contained *Bluetooth* product and requires only power to implement full *Bluetooth* communication. The integrated, high performance antenna together with the RF and Base-band circuitry provides the *Bluetooth* wireless link and the UART interface provides a connection to the host system.

The variety of interfaces and the AT command set allow the Embedded Intelligent *Bluetooth* Module to be used for a wide number of short range wireless applications, from simple cable replacement to complex multipoint applications, where multiple radio links are active at the same time.

The complexity and flexibility of configuration are made simple for the design engineer by the integration of an extremely comprehensive set of AT commands, supplemented with a range of "S" registers which are used for non-volatile storage of system parameters. These are fully documented in the "Blu2i AT Command Reference Manual".

6.1 Interfaces

6.1.1 UART interface

UART_TX, UART_RX, UART_RTS and UART_CTS form a conventional asynchronous serial data port with handshaking. The interface is designed to operate correctly when connected to other UART devices such as the 16550A. The signalling levels are nominal 0V and 3.0V and are inverted with respect to the signalling on an RS232 cable. The interface is programmable over a variety of baud rates; no, even or odd parity. The default condition on power-up is pre-assigned in the external Flash. Two-way hardware flow control is implemented by UART_RTS and UART_CTS. UART_RTS is an output and is active low. UART_CTS is an input and is active low. These signals operate according to normal industry convention.

By writing different values to the relevant S register the UART_RI can be continuously polled to detect incoming communication. The UART_RI signal serves to indicate incoming calls.

UART_DSR is an active low input. It should be connected to DTR output of the host. When the module is running in high speed mode (See definition for S Reg 507), this pin should be asserted by the host to ensure connection is maintained. A de-assertion is taken to mean that the connection should be dropped, or an online command mode is being requested.

The module communicates with the customer application using the following signals:

RS-232

Port /TXD @ application sends data to the module's UART_RX signal line

Port /RXD @ application receives data from the module's UART_TX signal line

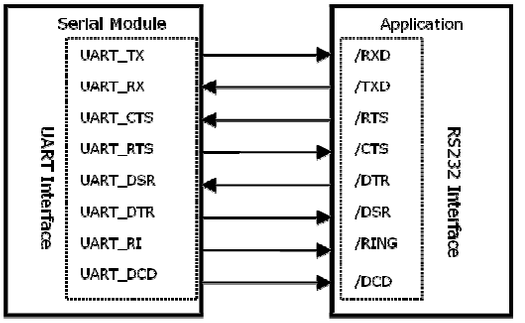


Figure 6.1 : UART interfaces

Note that the serial module output is at 3.0V CMOS logic levels. Level conversion must be added to interface with an RS-232 level compliant interface.

6.1.2 GPIO Port

Eight lines of programmable bi-directional input/outputs (I/O) are provided that can be accessed either via the UART port, or Over The Air (OTA) from a second *Bluetooth* unit. These can be used as data inputs or to control external equipment. By using these in OTA mode, an embedded *Bluetooth* Serial module can be used for control and data acquisition without the need for any additional host processor. A further line can be used as an input.

Each of the GPIO ports can be independently configured to be either an Input or Output. A selection of ports can be accessed synchronously.

The ports are powered from V_{CC}. The mode of these lines can be configured and the lines are accessed via S Registers in the range 623 to 629.

Low latency I/O can be accessed by using Ezurio's I/O via an enhanced inquiry process.

6.1.3 PCM CODEC Interface

PCM_OUT, PCM_IN, PCM_CLK and PCM_SYNC carry up to three bi-directional channels of voice data, each at 8ksamples/s. The format of the PCM samples can be 8-bit A-law, 8-bit µ-law, 13-bit linear or 16-bit linear. The PCM_CLK and PCM_SYNC terminals can be configured as inputs or outputs, depending on whether the module is the Master or Slave of the PCM interface.

In applications where the PCM master cannot supply a clock signal, the module can be configured to generate a clock signal on this GPIO: PCM_SLVCLK. Please contact an Ezurio FAE for further details.

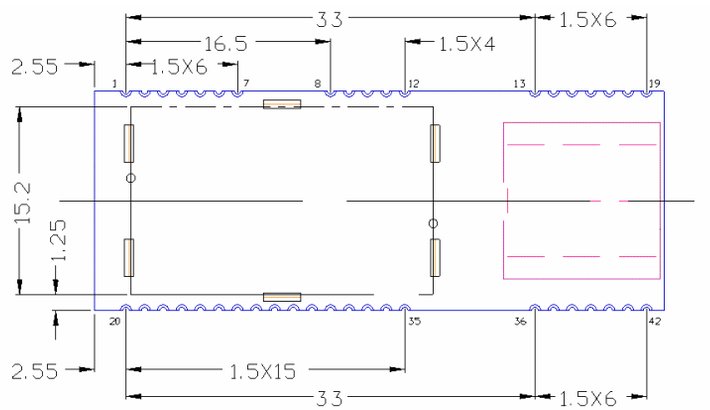
The Module is compatible with the Motorola SSI TM interface and interfaces directly to PCM audio devices including the following:

6.1.3.1 Compatible Codec Chips

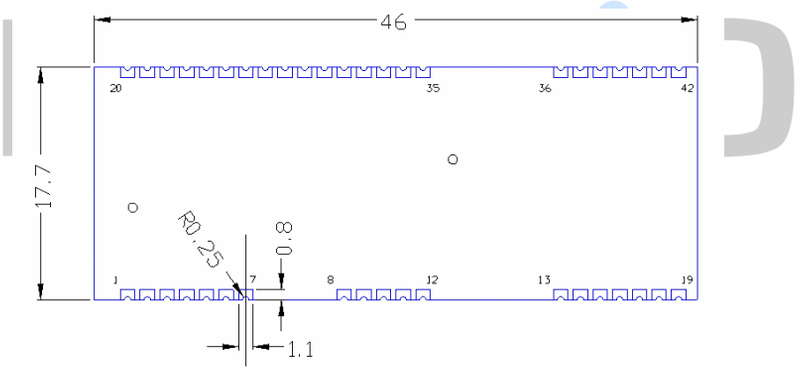
- Qualcomm MSM 3000 series and MSM 5000 series CDMA baseband devices
- OKI MSM7705 four channel A-law and µ-law CODEC
- Motorola MC145481 8-bit A-law and µ-law CODEC
- Motorola MC145483 13-bit linear CODEC

13. Physical Dimensions (all dimensions in mm)

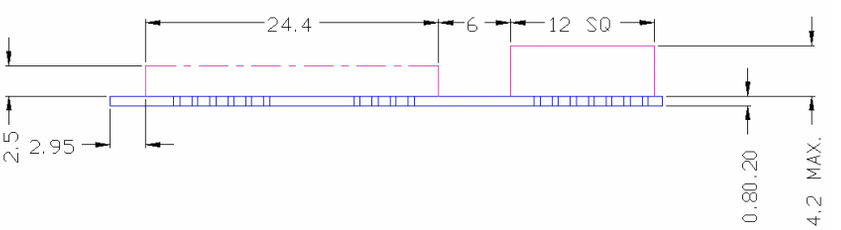
13.1 Top View



13.2 Bottom View



13.3 Side View



AT Command Set

Latest Development Firmware – Version 2.7.0

Contents

Audio Gateway	1112
WAP	1113
WAP_CLIENT	1114

2.2.6 ATD<U><Y><bd_addr>,<ServiceName> {Make Connection}

Make a connection to device with Bluetooth address <bd_addr> and profile specified via S Reg 101 AND which has a service name starting with the string <ServiceName>. The ServiceName parameter is a string delimited by ".

If <U> is not specified, then authentication is as per register 500, otherwise the connection will be authenticated.
If <Y> is not specified, then encryption is as per register 501, otherwise the connection will have encryption enabled.

The timeout is specified by S register 505.

Response: <cr,lf>**CONNECT 123456789012**<cr,lf>
Or <cr,lf>**NO CARRIER**<cr,lf>

2.2.7 ATD<U><Y>L {Remake Connection}

Make a connection with the same device and service as that specified in the most recent ATD command. The <UY> modifiers are optional. An error will be returned if the 'L' modifier is specified AND a Bluetooth address.

If both 'L' and 'R' modifiers are specified then an error will be returned.

Response: <cr,lf>**CONNECT 123456789012 AE**<cr,lf>
Or <cr,lf>**NO CARRIER**<cr,lf>

2.2.8 ATD<U><Y>R {Make Connection to peer specified in AT+BTR}

Make a connection with the device address specified in the most recent AT+BTR command. The service is as specified in S Register 101. The <UY> modifiers are optional. An error will be returned if the 'R' modifier is specified AND a Bluetooth address.

If both 'R' and 'L' modifiers are specified then an error will be returned.

Response: <cr,lf>**CONNECT 123456789012 AE**<cr,lf>
Or <cr,lf>**NO CARRIER**<cr,lf>

2.2.9 ATEn {Enable/Disable Echo}

This command enables or disables the echo of characters to the screen. A valid parameter value will be written to S Register 506.

E0 Disable echo.
E1 Enable echo.

All other values of n will generate an error.

S632	n/a	0..65535	When GPIO2 is configured as an input, low to high transitions are counted. There is no software debouncing. External RC circuit may be required. The counter wraps to 0 when it overflows beyond 65535.
S641	n/a	0..65535	As per 631, but the action of reading the value will reset the count to 0.
S642	n/a	0..65535	As per 632, but the action of reading the value will reset the count to 0.
S701	n/a	0..65535	Read/Write to Analogue Line 0, when reading value is returned in decimal
S702	n/a	0..65535	Read/Write to Analogue Line 1, when reading value is returned in decimal
S711	n/a	0000..FFFF	Read/Write to Analogue Line 0, when reading value is returned in hexadecimal
S712	n/a	0000..FFFF	Read/Write to Analogue Line 1, when reading value is returned in hexadecimal
S721	0	0	Set direction of Analogue Line 0
S722	0	0	Set direction of Analogue Line 1
S1001 to S1010		0.. 2^32	10 General Purpose 32 bit Registers for use by host. These are stored in non-volatile memory.

2.2.14 ATSn? {Read S Register Value}

This will return the current value of register n.

For recognised values of n
Response: <cr,lf>**As Appropriate**<cr,lf>**OK**<cr,lf>

For unrecognised values of n
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.15 ATSn=? {Read S Register – Valid Range}

This will return the valid range of values for register n.

For recognised values of n
Response: <cr,lf>**Sn:(nnnn..mmmm)**<cr,lf>**OK**<cr,lf>

For unrecognised values of n
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.16 ATX<string> {Send Data in Local Command and Connected Mode}

This command is used to send data to the remote device when in local command and connected mode.

The parameter <string> is any string not more than 24 characters long. If a non-visual character is to be sent then insert the escape sequence \hh where hh are two hexadecimal digits. The 3 character sequence \hh will be converted into a single byte before transmission to the peer.

Response: <cr,lf>**OK**<cr,lf>

Please refer to the “Power Consumption” chapter in the relevant blu^{2i} device user guide, for more detailed information of power usage.

The new values are NOT updated in non-volatile memory until the AT&W command is sent to the blu^{2i} device.

Response: <cr,lf>**OK**<cr,lf>
Or
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.19 AT&F* {Clear Non-volatile Memory}

The AT&F* variant of the command installs values in S registers as per command AT&F4 and then all other user parameters in non-volatile memory are erased. This means that the trusted device database is cleared, and so are parameters related to the following commands:- AT+BTR, AT+BTN, AT+BTS.

Response: <cr,lf>**OK**<cr,lf>
Or
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.20 AT&F+ {Clear Non-volatile Memory}

This command erases all user parameters in non-volatile memory except S Registers 520 to 525. This means that the trusted device database is cleared, and so are parameters related to the following commands:- AT+BTR, AT+BTN, AT+BTS.

Response: <cr,lf>**OK**<cr,lf>
Or
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.21 AT&W {Write S Registers to Non-volatile Memory}

Writes current S Register values to non-volatile memory so that they are retained over a power cycle.

Response: <cr,lf>**OK**<cr,lf>
Or
Response: <cr,lf>**ERROR nn**<cr,lf>

2.2.22 AT+BTAn {Control Audio Channel}

Once a Bluetooth connection is active, **and assuming the peer device is an EZURIO blu^{2i} device**, this command is used to start/stop a SCO channel which will connect the PCM interfaces of the two peer devices. This means that if a codec is attached to the PCM pins, then 2-way audio can be established.

+BTA0 Switch off the channel.
+BTA1 Switch on the channel.

On receipt of the command, the following response immediately follows.

Response: <cr,lf>**OK**<cr,lf>

Response: <cr,lf>**ERROR 14**<cr,lf>

ERROR RESPONSE
A Bluetooth inquiry process is such that for a single inquiry request a device could respond many times. To ensure that an address is sent to the host only once for a particular AT+BTI, an array of addresses is created at the start of each AT+BTI and is filled as responses come in. This array of addresses is stored in dynamic memory and as such if the memory allocation fails then the inquiry procedure is aborted and in that case an error response is sent to the host.
*To clarify, a single AT+BTI will **never** return the same Bluetooth address more than once, but as long as the responding device is active, all AT+BTI commands will always return it.*

Response: *<cr,lf>**ERROR 27**<cr,lf>*

2.2.35 AT+BTIV<devclass> {Inquire}

As per AT+BTI but the response includes the device class code for all inquiry responses. Please refer to the 'ERROR RESPONSE' note in the description for AT+BTI<devclass>.

Response: <cr,lf>**12346789012,123456**
<cr,lf>**12345678914,123456**
<cr,lf>**OK**<cr,lf>

2.2.36 AT+BTIN<devclass> {Inquire}

As per AT+BTI but the response includes the device class code and friendly name for all inquiry responses. Please refer to the 'ERROR RESPONSE' note in the description for AT+BTI<devclass>. The friendly name strings are in UTF-8 format as per the Bluetooth specification.

Response: <cr,lf>**12346789012,123456,"EZURIO AT DONGLE 1"**
<cr,lf>**12345678914,123456, "EZURIO blu2i RS232"**
<cr,lf>**OK**<cr,lf>

Note: Many releases of firmware will return the product name as TDK, e.g.

Response: <cr,lf>**12346789012,123456,"TDK AT DONGLE 1"**
<cr,lf>**12345678914,123456, "TDK blu2i RS232"**
<cr,lf>**OK**<cr,lf>

We strongly recommend that any software implementation that uses this command should check for both TDK and EZURIO to ensure backwards and forwards compatibility.

2.2.37 AT+BTK=<string> {Set Passkey}

This command is used to provide a passkey when **PIN? 12345678** indications are received asynchronously. If a pairing is not in progress then the pin is written to non-volatile memory for future use. Specifying an empty string deletes the key from the non-volatile memory.

The string length must be in the range 0 to 8, otherwise an error will be returned.

Response: <cr,lf>**OK**<cr,lf>

2.2.38 AT+BTM<bd_addr> {Set Incoming Peer Address}

This command is used to store a peer address for incoming connections in non-volatile memory. A value of 000000000000 has the special meaning of invalid peer address.

When S register 512 = 3, 4, 6 or 7 then it will wait for an incoming connection from the peer address specified. If the peer address is not 000000000000, then it waits for a connection from the specified master, otherwise will connect to anyone.

Response: <cr,lf>**OK**<cr,lf>

2.2.39 AT+BTM {Delete Incoming Peer Address}

This command is used to delete the peer address previously stored using AT+BTR<bd_addr>.

Response: <cr,lf>**OK**<cr,lf>

2.2.40 AT+BTM? {Read Incoming Peer Address}

This command is used to display the peer address stored in non-volatile memory, used to put the module in pure cable replacement mode.

Response: <cr,lf>**12346789012**
<cr,lf>**OK**<cr,lf>

If the location is empty the response is as follows.

Response: <cr,lf>**00000000000**
<cr,lf>**OK**<cr,lf>

2.2.41 AT+BTN=<string> {Set Friendly Name in Non-volatile Memory}

This sets the default friendly name of this device as seen by other devices. It will be stored in non-volatile memory. Use AT+BTF to make the name visible to other devices. Use AT+BTN? To read it back. An empty string ("") will delete the string from non-volatile memory which will force the default name to be used.

Response: <cr,lf>**OK**<cr,lf>

2.2.42 AT+BTN? {Read Friendly Name from Non-volatile Memory}

Read the default friendly name from non-volatile memory.

Response: <cr,lf>**"My FriendlyName"**<cr,lf>
<cr,lf>**OK**<cr,lf>

2.2.43 AT+BTO<devclass> {Open and make Unit Detectable}

After power up and ATZ, this command is sent so that RFCOMM is initialised and opened and the service name as specified in AT+BTN is exposed via the SDP registry.

The <devclass> value specifies an optional fixed length hexadecimal device class code. If it is not specified, then the device class code is taken from S Register 515.

For this command to be effective, S Register 512 must be set to 0.

Response: <cr,lf>**OK**<cr,lf>

2.2.44 AT+BTP<bd_addr> {Enable Cautious Page/Inquiry Scanning}

Enable page scanning and wait for a connection from device with Bluetooth address <bd_addr>. If the specified address is 000000000000 then incoming connections are accepted from any device, is as per AT+BTP without an address. Inquiry scanning is also enabled.
This command also has variants which allow authentication and encryption to be explicitly specified. For example:-

AT+BTPU123456789012
AT+BTPY123456789012
AT+BTPUY123456789012
AT+BTPYU123456789012

Response: <cr,lf>**OK**<cr,lf>

2.2.45 AT+BTP {Enable Promiscuous Page/Inquiry Scanning}

Enable page scanning and wait for a connection from any device. Inquiry scanning is also enabled. Authentication and Encryption is as per S registers 502 and 503.

Response: <cr,lf>**OK**<cr,lf>

2.2.46 AT+BTPU {Enable Promiscuous Page/Inquiry Scanning}

Enable page scanning and wait for a connection from any device. Inquiry scanning is also enabled. Authentication is enabled and encryption is disabled.

Response: <cr,lf>**OK**<cr,lf>

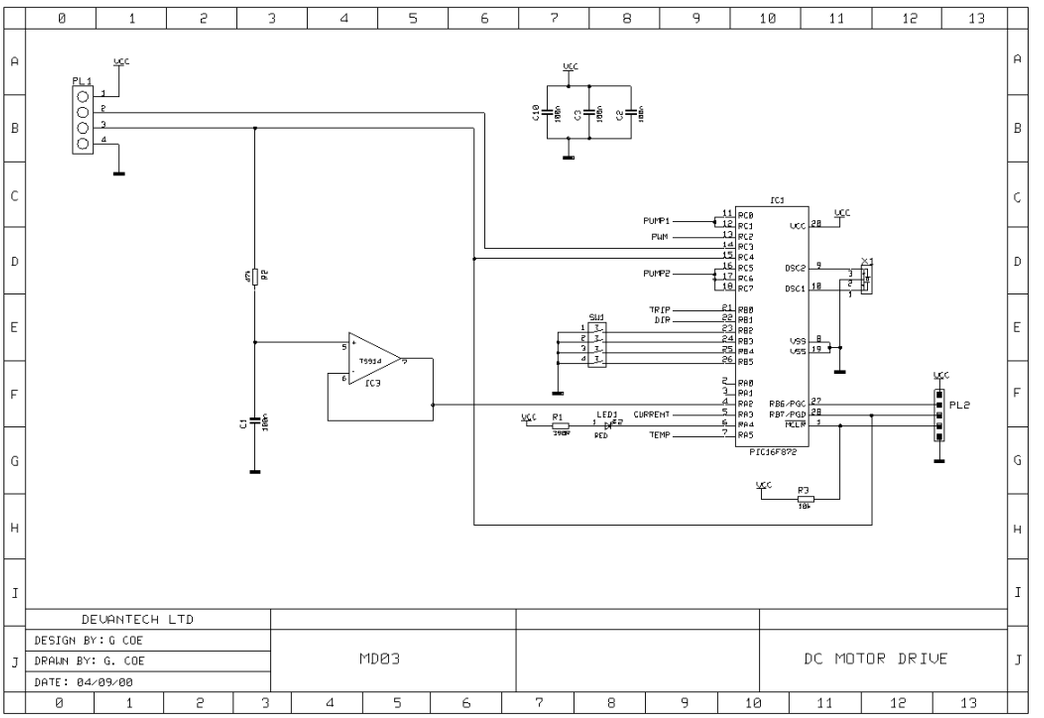
2.2.47 AT+BTPY {Enable Promiscuous Page/Inquiry Scanning}

Enable page scanning and wait for a connection from any device. Inquiry scanning is also enabled. Authentication is disabled and encryption is enabled.

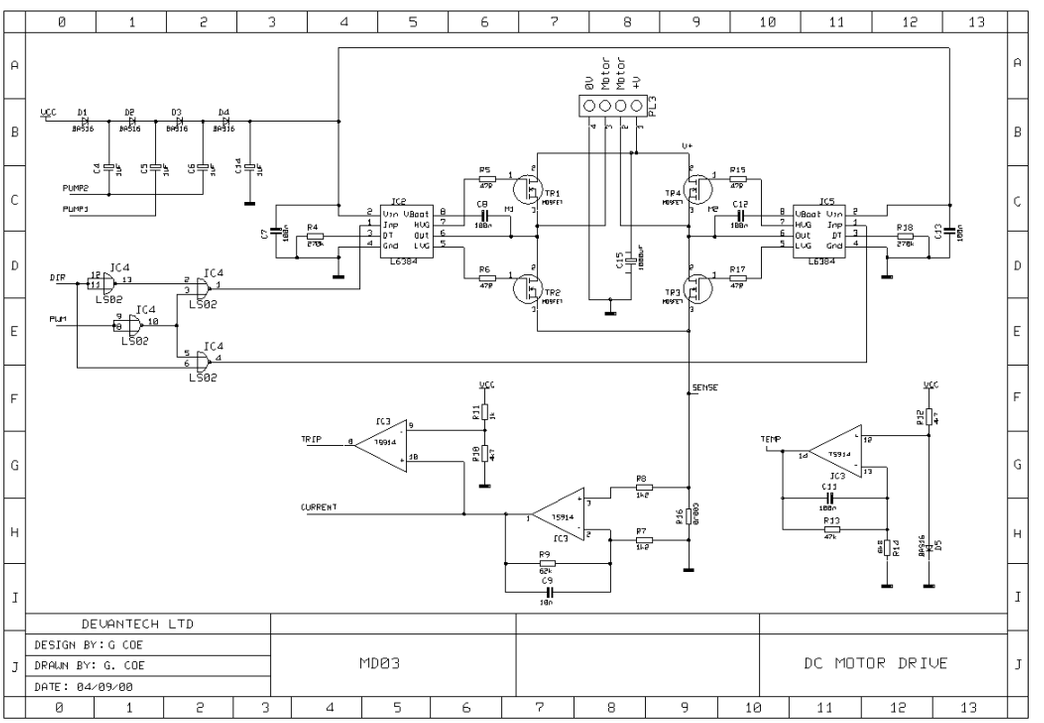
Response: <cr,lf>**OK**<cr,lf>

2.2.48 AT+BTPUY {Enable Promiscuous Page/Inquiry Scanning}

Enable page scanning and wait for a connection from any device. Inquiry scanning is also enabled. Authentication and encryption are both enabled. The order of U and Y is not significant.



DEVANTECH LTD	
DESIGN BY: G COE	MD03
DRAWN BY: G. COE	DC MOTOR DRIVE
DATE: 04/09/00	



DEVANTECH LTD	
DESIGN BY: G COE	MD03
DRAWN BY: G. COE	DC MOTOR DRIVE
DATE: 04/09/00	